CHALMERS

Contextual Reasoning based Mobile Recommender System

Master of Science in Applied Information Technology Thesis

- *Intelligent System Design*

ABHIROOP GUPTA
Department of Applied Information Technology
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, 2009
Report No. 2011:087
ISSN: 1651-4769

Contextual Reasoning based Mobile Recommender System

Abhiroop Gupta

Examiner: Claes Strannegård

Chalmers University of Technology
University of Gothenburg
Department of Applied IT
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 6032

Department of Applied Information Technology
Göteborg, Sweden September, 2011

Preface

This Master’s Thesis was carried out at Ericsson Research, Kista, Sweden from February of 2011 to July of 2011. The thesis is a prerequisite final work for a Master’s Degree in Intelligent Systems Design at Chalmers University of Technology, Göteborg, Sweden. The industrial supervisor at Ericsson Research was Mr. Simon Moritz and academic supervisor at Chalmers University of Technology was Mr. Claes Strannegård.

Abstract

With the ever increasing popularity of Smartphone and reducing charges for device and data Mobile applications users are estimated to quadruple in the next three years. With the increase in the number of users Applications in the market are increasing by thousands every day. The Users are flooded with so much of choices that it is hard for them to find appropriate and Suitable apps. Recommender systems can aid the users in discovering new applications in a personalized manner. The purpose of this thesis is to investigate how to enhance the recommendations to a user in the process of discovering new mobile applications by better utilization of context data available through various sensors in a modern day Smartphone. The work of the thesis is divided into three phases where the aim of the first phase is to study related work and related systems to identify promising concepts and features. During the second phase, a prototype system is designed and implemented. The outcome and result of the first two phases is then evaluated and analyzed in the third and final phase. The prototype system integrates an existing mobile app recommender system to add the features of context awareness and context reasoning. The major draw of the thesis is to suitably define context and situations of interests and model them appropriately. Learning techniques are applied to learn these models using the context data generated by the user. The derived situation provides an added parameter in the process of information filtering in a personalized manner.

Table of Contents

Table of Figures

Introduction

Just about everyone loves the convenience of having their own personalized interests delivered to them without much work. That’s exactly what recommendation engines do for their users. With the advent and popularization of e-commerce recommender systems came in forefront, these engines remember your preferences and make suggestions for you based on your history. The field of Mobile apps has become so flooded with apps in the recent times that Recommendation engines have become the need of the hour for Smartphone users to aid them discover apps of their personal interest and rightly so it has gained interests among the researchers and system developers.

With over 440,000 applications available for the iPhone and over 275,000 available for the Android platform, there is an apparent information overload emerging in the domain of mobile applications. The fact that these applications usually are browsed and downloaded from the mobile device, with its smaller screen, makes this information overload even more intense.

Recommender engines uses a specific type of information filtering system technique that attempts to recommend information items (movies, TV program/show/episode, video on demand, music, books, news, images, web pages, scientific literature such as research papers etc.) that are likely to be of interest to the user. Typically, a recommender system compares a user profile to some reference characteristics, and seeks to predict the 'rating' that a user would give to an item they had not yet considered. These characteristics may be from the information item (the content-based approach) or the user's social environment (the collaborative filtering approach).

The purpose of the thesis is to identify techniques and algorithms that work on the generated data of the Smartphone App users to predict the context of the user and to build a prototype of a system which can achieve this task and thereby help the underlying Recommender System to enhance the recommendations.

In our approach we tried to model the relationship between context and situation effectively so that the application can map the sensor originated context data to predict the situation of the user with some degree of certainty. The outcome of the contextual reasoning adds an extra level of information filtering which can be achieved on the User data to recommend him with apps.

The project was initiated at Ericsson Research last year when a mobile recommender system was developed based on an underlying Ericsson recommendation engine developed initially to recommend media items to the users. The Recommender system also had to capability to filter information based on “Apps popular around you” to address the cold start problem.

The underlying Recommender engine incorporates the techniques of Collaborative Filtering to compute the recommendations for a User/Item. Collaborative filtering (CF) is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. In Recommendation engines it can be applied as a method of making automatic predictions (filtering) about the interests of a user by collecting taste information from many users (collaborating). The underlying assumption of CF approach is that those who agreed in the past tend to agree again in the future. It is to be noted that these predictions are specific to the user, but use information gleaned from many users.

Since the mid-1990s, recommender systems have become an important research area and several systems have been developed and deployed. Systems such as Amazon.com have presented recommendations on the form “people who bought this, also bought that” to users for over a decade. A system which minimizes the information overload in a personalized way would be beneficial not only for the end user, but for every actor in the mobile ecosystem; end users, developers, operators and retailers. The utility of recommender systems was reinforced when Amazon.com reported a 35% jump in sales the following year of their deployment of recommender system[2].

With the advent and increased use of social networks in recent years, it is possible to filter products and services based on what a user’s friends are consuming. Users could share or recommend applications to each other, a digital equivalent to the analogue word-of-mouth, or a user could be allowed to browse her friend’s applications. These methods are increasingly becoming popular in recommendation systems. This method results in more personalized recommendations but may introduce privacy issues and not everyone is keen on online social networks.

With technology been moving beyond desktop computer to everyday devices, the field of pervasive computing came to the forefront of research in the field of computer science. Many concepts developed by the field of pervasive computing also find their place in other fields. One such concept is Context and Context Awareness. A.K. Dey [5] in his work defines Context as “*Any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.”* He further says “*A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task*”.

The concepts of Context Awareness is applied thereafter in many fields apart from pervasive computing including Health care, UI design, Search Engines, Advertisement, Recommendations etc. to name a few. It has been seen that Context Awareness aids in better interaction between the User and the System. Gartner in its Report[1] notes that “By 2015, context will be as influential to mobile consumer services and relationships as search engines are to the Web.”

Objectives

The Goal of the Thesis is to investigate, find and implement suitable features that can be added to the existing Recommender System for Android Apps. The techniques being implemented should fall under certain predefined boundaries namely:

- Good enough App Recommendations should be provided by the system
- The system should be capable of providing recommendations from the day one.
- The number of inputs required by the system from the user should be kept as minimum as possible.
- Contextual data generated by the user should be utilized as effectively as possible.
- The system must possess some quality to learn about the choice of the user.
- The solution should preferably consist of a recommender system and possibly other filtering solutions.

An in depth study of the Android market has to be performed to analyse the Android App Market, Various other App Recommender System available to the user and their features. Suggestions for modifications and enhancements to a recommender engine previously developed at Ericsson AB should be presented.

Problem

After an in-depth study performed on the Android market and State of the current research in the field of Recommender Systems, a set of features were identified to be implemented on the underlying Recommender System and thus the problem was defined.

The general problem addressed in this thesis report is addressing context awareness and contextual reasoning in mobile app recommender systems. Defining proper models for Context Reasoning, using the models effectively, predicting about the users situations and effectively utilizing these deductions to better the recommendations are the main challenges that are addressed in this thesis. The effort in terms of time and cognitive work a user puts in to finding interesting and relevant applications should be as low as possible. This problem was divided in to sub-problems, namely; collecting and storing user's sensor originated data; Contextual Reasoning around the Sensor originated data; and utilizing the same for an added level of information filtering by the Recommender System. These sub-problems are described in greater detail in the following sections.

Collecting sensor originated data

In order to form the input data for Context Reasoning the System must collect data about the Users environment using various sensors available in a modern day Smartphone. An example of such Sensors can be GPS data telling the Location of the User, Accelerometer to judge the motion of the User, Microphone to analyze noise around the user etc. to name a few.

Contextual Reasoning

The heart of the current work on the Recommender System is adding a Contextual reasoning engine. The Reasoning engine takes the various sensor originated data as context inputs. It enhances the quality of the input data by enriching it with data from Map APIs, GIS, Social Network etc. It then uses the context history of the user to predict the situation of the user through different machine learning algorithms/rule based systems. It also performs all necessary steps in between including Outlier Removal, Feature Extraction, Classification etc.

Effective Use of derived knowledge in the Recommender System

The general idea of filtering is to obtain a subset of items based on some criteria. In the domain of this thesis, filtering applications is the process of retrieving a subset of applications of high interest and relevance to the user based on the current situation of the User. The resulting subset of applications should be sorted to make applications at the top of the result more likely to be interesting to the user.

Presentation

Once the system knows which applications to recommend to a user, it must be able to present them. Presentation in this case does not only involve visualization such as the layout of the graphical user interface but also includes the available metadata for applications, the notion of correlation between applications, the actions one can take on applications, and how this affects the user's workflow.

Hypotheses

A set of hypothesis were formulated initially after studying the current system which some of which will be verified in the end if the scope of proving the hypothesis falls under the purview of the formulated problem and its solution. The hypotheses are presented in this section with their classification.

Towards expanding the context and addressing cold start

- **Hypothesis 1:** Data available in the Android OS can be effectively utilized to analyze patterns. Example of such data can be Network Operator Name, Sim Operator Name, Cell Location, Phone Number, whether in roaming etc.
- **Hypothesis 2:** User Demographics information either collected via the Mobile Operator or collected from the User directly from the recommendation app can be utilized effectively to analyze patterns.
- **Hypothesis 3:** Content of the metadata of the apps can be effectively utilized to find a list of relating apps and generate the similarity measure of the apps.
- **Hypothesis 4:** Making it optional to provide the demographics data for the user and providing them with statistics of how the data influences the recommendations will reduce the concerns of the user in providing personal information.
- **Hypothesis 5:** Providing users with explicit feedback options, e.g. Like/Dislike, Agree/Disagree buttons can help in better User Modeling.

Towards expanding to Social Network, blogs etc

- **Hypothesis 6:** Providing the user with an option to share interesting apps on social networking sites will open new source of generating recommendations based on what apps the user's friends are sharing etc.
- **Hypothesis 7:** Unstructured data from the web, blogs etc can be a potential source of metadata of the apps.
- **Hypothesis 8:** Semi Structured data on twitter can be utilized effectively to discover relations between apps.

Towards diversifying the recommendations

- **Hypothesis 9:** List of recommendations can be diversified based on recommendations from different classes of relating apps.
- **Hypothesis 10:** The recommendation engine can adapt to the amount of recommendations that is provided for different classes of recommendations based on the user's actions over a period of time dynamically.
- **Hypothesis 11:** The list of recommendations can be filtered in multiple levels to achieve diversity.

Scope

In this section, the scope of the thesis is presented along with a brief description of what will be evaluated and how. The section defines the limitations and focus of the project. The actual evaluation is presented in 6.

Scope of Hypotheses

A set of hypotheses are drawn after an initial study of the system some of which will be verified or rejected based on argumentation on the behavior and functionality of the system. The unverified ones will be presented in the future work sections if they are found to be promising after the through background study of the system. Some hypotheses may easily be verified by assuring that a part of the system works, while others may require a qualitative study and expert opinion in the form of a discussion.

Adding features to existing App Recommender System

An existing recommender system for Android apps is modified and used during the project. Based on the enhancements done to the system a discussion is presented on the new features and its utilization of the existing framework to enhance the process of recommendation.

Focus

The focus of this thesis has mainly been concentrated in the topics of (a) App recommendations techniques & concepts, (b) Context and consumption collection and (c) Context Utilization. The main problem addressed in the thesis the problem of Contextual reasoning and its utilization in the process of Recommendation.

Limitations

The focus of this thesis has not been towards performing any usability tests for the recommendations or any quantitative study of the recommender engine due to constraint in data and time.

Background Research

Formal Research on recommendation engines for internet started in early 1990's at Xerox PARC in response to the overwhelming number of emails. They created a system named "TAPESTRY"[3] in which users could filter information from all incoming streams including e-mails, news articles etc. It used both content based filtering and collaborative filtering. In the last 10 years with the penetration of the internet into our day to day business, recommendation engines were seen as a big way to boost the business as a result a lot of research work has been done in the field. In 2007 the Netflix Prize, a contest with a dataset of over 100 million movie ratings and a grand prize of $1,000,000, energized the search for new and more accurate algorithms and many new algorithms were developed.

Recommender Systems

Recommender systems were developed to overcome the problem of information overload by aiding users in the search for relevant information and helping them identify which items (e.g. media, product, or service) are worth viewing in detail. This task is also known as information filtering. This Section will introduce some recommender system techniques and approaches and give an overview of mobile recommendations.

Techniques

Recommender systems do the information filtering by predicting whether a user will like or dislike an item. This prediction is based on the user’s explicit and implicit ratings/preferences, other users’ ratings, and user and item attributes. For example, a music recommender could make use of implicit user data (e.g., Sven bought the Beatles’ White Album), explicit data (e.g., Sven rated Neil Young 4 out 5), user demographics (e.g., Sven is male), and item attributes (e.g., Nirvana is labelled as Grunge and Rock) to make recommendations.

Content-based

Recommendations can be based on the content of items, comparing the content of previously liked items with the content of unseen items and recommending similar ones. This approach is referred to as the content-based (CB) recommendation method. For example, a system recommending movies would analyze the movies a user likes to find out what they have in common in terms of content, i.e. actors, directors, genres, et cetera. This information will constitute the user’s preferences which are used to find movies with a high degree of similarity to the liked ones.

The major drawback of the content-based approach is its inability to identify qualities of items which are not machine readable or understandable. Humour and visual appeal are examples of such qualities. Content-based filtering also suffers from the *new user problem*, i.e. the user has to rate a sufficient number of items before the user’s preferences can be understood. Content-based recommender systems also require attribute and feature data of items, and this data may be difficult to collect.

Collaborative Filtering

Collaborative Filtering (CF) recommender systems, on the other hand, recommend items to a particular user based on how other users have rated items. A movie recommender system would find peers, users who have similar rating patterns to the user receiving recommendations. The movies with the highest ratings according to the peers, and which the user has not yet seen, would then be recommended. This approach is called User-Based Collaborative Filtering. There is also Item-Based Collaborative Filtering in which the items a user has rated are compared to all other items in terms of user ratings, the most similar ones with the highest average rating are then recommended.

Collaborative Filtering systems suffer from the new user problem, i.e. a new user in the system does not have enough consumption history for the system to find an overlap in ratings with other users. Collaborative Filtering also suffers from the new item problem which causes new items to be ignored (i.e. not recommended) until a substantial number of users have rated the item [2].

Social

Utilizing a user's social network, digital or not, to produce recommendations is in most systems a matter of trust. Trust is a Social Network based recommender system's equivalent to user similarity of collaborative filtering. If a user's trusted connections are known, producing recommendations is a matter of identifying the trusted connections' highly rated items.

The difficulty is, just like in the collaborative filtering case, to identify and rank the user's trusted connections (or neighbourhood). But for the sake of argument, one could identify a user's trusted connections as all her friends and friends of friends.

Hybrid

In an effort to avoid the limitations of Collaborative Filtering and Content-based systems, hybrid approaches which combine the two have been proposed [2]. As an example, such a system could implement the two methods separately and then combine their results. A hybrid approach could also incorporate any other method, such as recommending items based on what has been consumed in locations close to the user.

Mobile Recommender Systems

Mobile devices, such as phones and PDAs, are evolving in to a major source of information [4]. In the past, many recommender systems have been developed for the desktop computer. However, these systems cannot be applied directly as an aid for mobile users since mobile recommender systems need to overcome many of the challenges generally present in the domain of mobile devices.

Usability and Interaction
Recommending on a mobile device introduces new challenges due to the qualities of the device itself and the context of its use. Compared to traditional desktop computers, a mobile device has a smaller screen and limited input capabilities [8]. The screen size of future devices is unlikely to improve as it is a necessity for the device's mobility. The users of mobile devices will also be in a different environment compared to the desktop. This environment is also likely to be changing continuously during use. Theoretically any context must be considered when designing the user interface and workflow of the application.

Possibilities
Mobile devices are not all about limitations, they also offer great possibilities. The physical location of a device, sensory data and such could prove an important and worthwhile source of information [4]. This type of data reveals the context in which the device is used at a given time. This contextual information could aid the recommendation process by providing more data to base recommendations on. It may for instance be wise to recommend travel related applications to a user who is located at an airport.

The problem in hand needed an in-depth study of the Apps Market and App Recommender Systems to be done as well as an in-depth study of the current research been taken up in the field by different researchers. In this section both of these are presented in detail.

Android Apps Market

The Android Apps market is growing leaps and bounds each day. The Users are flooded with so much of information that it is just next to impossible for them to discover interesting apps. Therefore the problem of information overload is more than ever before and hence recommendation engines have their unique need in this area. During the project was undertaken to identify different android apps markets and currently available app recommender systems. Google's Android Market is not the only source of Android Apps however there exists 17 different markets out of which 6 are dedicated to Android Apps while 11 are shared markets, which are listed in the table in Appendix I.

Available App Recommender Systems

Currently there are some App recommender systems available in the market from different sources and companies which were studied in detail during the project to identify their features and the taste of the users. It was seen that Social Recommendations are the trend of the day as many of these systems use integration with social network to generate recommendations as their key feature. In this section some of the App Recommender Sytems studied are discussed in detail.

AndSpot (Market)

AndSpot have incorporated a social network in their Android Market

Features of Interest

- Have little characters to choose from giving the users option to describe their mood and personality.
- View what your friends are installing, rating and discussing
- Add friends and view their profiles
- Send receive messages from your friend in the application
- Discuss about apps in the client application(A blog for every application)
- Share apps in facebook or twitter.

Critiques

- It's more of sharing and seeing what other people are sharing than recommending.
- It is based on collaboration rather than collaborative filtering.

Appolicious

Appolicious Android Apps is the place to discover and share the latest Android apps through social recommendations as well as reviews from users and their editorial team.

Recommendations based on

- What apps you already own
- The apps the people you follow own
- What kinds of apps you're interested in when you join Appolicious Android apps.

Appaware(Recommender App)

The AppAware helps to find new Android apps that are currently installed by the community. User can also Share installations and benefit by discovering apps that are installed by friends, many users (the trend!) and around your location.

Features of interests:

- Displays and updates the list of application (what users around you are doing) in real-time in the client app.
- Lets the user to tag applications and search by tags
- Lets the user to find his friends who are using appaware by accessing Facebook friend list/Gmail contact list/twitter following and then get updated with what friends are doing.

Appreciate

Appreciate is another cool app recommender for android. This one also tries to capitalize on social network to generate recommendations

Features of Interests:

- Updates the user with a list of application what your friend has installed/uninstalled.
- Let's you set your preferences for recommendations Apps/Games, Free/paid, New/All-time
- Let's you login with your Facebook/ a nickname account.
- Recommendations are generated based on what you have installed according to their site but I think they use some location data also as when I used there were quite a few apps useful and applicable for Sweden and Stockholm.
- The latest activity region features what other users are doing in the system. It shows new users who have joined, user x has installed app y etc.

Critique

- What was striking was when you click on a user in the latest activity region it opens up with a page in which all the apps installed by the user are displayed. This might pose some serious privacy concern for some users.

AppJoy

AppJoy helps discovering interesting Android applications. Unlike other tools that make recommendations based on applications' download popularity, AppJoy uses the actual usage metric to rank applications and makes personalized recommendations based on users own application usage.

Recommendations are divided into 5 lists namely

- My Recommendations (based on users application usage)
- Location based search(what is popular in your region)
- Most Recent Applications / Top Scored Applications(description not given but I think based on the sum of user ratings given to the application. Also shows the average rating of the application)

Appazaar

Although the available features on the current App from Appazaar is limited their prototype system is described in a paper [6] where they discuss a prototype system. The prototype is realized as a widget, the screenshot of which is shown below.

In the prototype system a logger is implemented to keep track of the running services and location of the device. The widget communicates with the server via HTTP requests, mainly to upload the recorded data and receive recommendations. The inference engine uses location and time as context information and builds contexts independent from users. A further discussion is presented in section 2.4.1 and 2.4.2 on some related articles as this work has been one of the main motivations for the project.

Summary

Most of the recommendation engine studied during the work tries to leverage the power of social network in improving their recommendations. 2.3.1, have their own social network to which its users are registered. 2.3.2, use social network to recommend apps to friends while in 2.3.4 a user can have their Facebook Login to log onto the system and show others what they have installed but poses serious privacy concerns. 2.3.4 also has social network integration but the unique feature they have is providing the users with real-time data of what is happening in and around them, what other users near the user are consuming right now. 2.3.5, also have location based recommendation but it is not real-time in nature and they also have results from classical collaborative filtering, presented to the users in two separate lists rather than weighting and merging them together. In this way they optimize the task of presenting the user with information about the source of recommendation. While in 2.3.6 is a recommendation system with tries to use contextual reasoning to build contexts independent from the users in order to provide the recommendations which became one of the driving motivations for this project.

Current State of Research of Recommender Systems and Context Aware Computing

From the early 90's the research on Recommender Systems have come a long way and various techniques have been applied by different researchers to improve recommendations and come up with new ways to generate recommendations. The aim has always been to make recommendations more personalized. Researchers are also trying to come up with algorithms which can not only generate personalized recommendations but also recommendation suitable for a particular user in a specific instant.

A lot of papers were studied in detail during the project to get a strong understanding of the different techniques being used and implemented currently. Although papers from all the domains where recommender systems have been applied are studied the focus of the study has been towards the domain of Context awareness, Mobile Applications and Services. In this section some key notes of the papers studied are presented.
Research directly related to presenting the concepts of context awareness includes some of the below mentioned works.

The term "Context Aware" was introduced by Schilt and Theimer and he refers it as location, identities of persons and objects nearby and changes to those. It is difficult to apply context by such definition by example and hence the [5] gives a formal definition of the same as:

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between the user and the system including the user and the system themselves."

"A system is context aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on user's tasks."

The authors of [5] also present a framework called "The Context Toolkit" which aims to aid the development of Context Aware Applications and is widely popular in the community.
[4] in their work titled "Reasoning in Context-Aware Systems" try to identify the main challenges in future applications that take into account the user's context and use some kind of context reasoning. The Author defines Context Reasoning as "deducing new and relevant information to the use of application(s) and user(s) from the various sources of context-data." The Authors argue that Context by nature is hierarchical in the sense that raw context data can be mapped to higher level context. The author stresses on the importance of Sensor data fusion in integrating data from multiple sensors (sources) in a reliable way, necessity to recognize new contexts, need for Context monitoring for the system to be proactive and the need for model monitoring in such systems.

Research in the application of context awareness includes some of the below mentioned work

[6] in their work titled "The Case for Context-collaborative Filtering in Pervasive Service environments" presented a high level design of a context sensitive collaborative filtering model for mobile services. They conceptualized a design of two approaches namely Context Aware user centric approach where applications are projected relevant to the context and Context Aware Collaborative which mixes context and collaborative filtering. The authors also in another of their work [7]

critically evaluate the design space and the different techniques that can be applied to capture different data about the user and their applicability which provided a good foundation for the work done under the scope of this thesis. [6] Presents a design for context collaborative filtering and aim to use context reasoning along with usage interpretation but didn't describe context reasoning in detail however in their paper for prototype implementation they used a tagging approach to collect data from the users on "What are you Just doing? Tag it!" to get input of users activity. However this again leads to problem (e).

M. Pryzybilski et al [8] defined a context reasoning framework. It is a generic framework developed that can be used by any context reasoning systems. However, it is difficult to induce situational knowledge in the framework. Moreover, as new sensors are added to the system each and every reasoning mechanism will have to be changed to incorporate the knowledge of the new sensor if the reading affects it.

D. Cheng [9] et al used in their work anonymous situations and unsupervised learning to train the system. However, unsupervised learning of context and situation need a lot of initial training data to provide recommendations from the beginning and that is an essential requirement in recommender systems.

[10] presents a system for Context-Aware Recommendations in the Mobile Tourist Application which is a good approach to build such systems. However, the application of the framework is limited to tourism domain.

A lot of work has also been done in the field of multidimensional context filtering [13][12][11]. In system where you have large number of contexts a multidimensional approach involving Matrix Factorization is not flexible enough to add contextual dimensions in a straightforward manner [7]. [12] Further describes a tensor factorization that overcomes this problem. However using only a tensor factorization it is difficult to explain to users the reason they received a particular recommendation.

[14] In their work discuss the importance of explanations in recommender systems and argue that they give better understandability and acceptance of the system, better control for the user and aids in building trust between the user and the system. Defining the situations also opened the path to collect optional feedback from the users about their correct situation if they feel the sensed situation is incorrect and help the system in learning about the user.

C. Davidsson and S. Moritz [15] in their work used location and time of the day as a context to enrich the personalization. While location, time etc. as context provides personalization, higher level of personalization can be achieved by reasoning around combinations of them rather than treating each individually.
Sholmo Berkovsky et al[16] use context reasoning using RDF/OWL format for User Modeling task which is a good way to model the rules in a rule based reasoning engine.

Thinking ahead: HTML5 Vs Android App

Since it is widely believed that everything in the future are moving towards HTML5 a study was also undertaken during the project to analyze the merits and demerits of keeping the recommender system client as an android app rather than moving to HTML5 app. The results of the study [6] are briefly discussed as follows:

- Android Apps provide a way to improve web app using the Android SDK Features. It is important to know before making the decision what features of the Android SDK can provide potential improvement to the web app.
- In the case of HTML 5 the application has a larger audience and a wider appeal and the web is likely to stay here for a long time unlike other platforms which continuously keep on changing.
- The strengths of Android native aps are as follows :
    - Strong Hardware integration
    - Apps integrate system features
    - Better inter app communication
    - Faster execution time
    - Ubiquitous nature
    - Rich in multimedia
- Current Android platform features of interest
    - Geo Services
    - Sensors
    - Inter process communications(Intents)
    - Background process
    - Full Database
    - Camera and Microphone access
- Strengths of HTML5
    - Support across multiple platforms.
    - Support across multiple device types(PC/Laptop/Mobile/Other)
    - Provides mechanisms to share data securely like O-Auth, Cross Origin Resource sharing and Cross Document Messaging etc. but the boundaries are porous.
    - HTML5 has the functionality to fetch Geo Location data but requires permission from the user and hence not every app will get it.
    - HTML5 supports orientation.
    - HTML5 supports Speech detection.
    - HTML 5 supports Idle detection.
    - Camera and Microphone access

Design

Goals

Research Goals

After formulating the problem to be addressed in this thesis three specific research goals were initially formulated focusing on two major issues: (1) Understanding various options available to the users in the domain of App Recommender System and analyzing the shortcomings of each and what can be done (2) To design such a system which can effectively take the advantages of information available in the mobile domain and overcome the challenges which are as follows.

- To analyze constraints and additional sources of information available in mobile domain.
- To design a flow based structure of a system which effectively utilizes the available information and overcomes the constraints.
- To analyses the benefit such a system brings in.

The three goals questions formulate a rather large and comprehensive set of research issues to be explored. The three goals involve a rather large number of subsequent second level questions and issues to be worked on. It is therefore not reasonable address all three research goals in the scope of this thesis. In the beginning of this thesis the focus was therefore delimited to the first and second question above. A problem was formulated to address the research goals of this thesis which is mentioned in section 3.2 in detail.

Industry Goals

Recommendation engines create paths to purchase by reducing the bounce rate and improving return on search marketing investment. From an industry perspective this research work is expected to contribute to an enhanced understanding of how a Contextual Reasoning based recommender system can be build and utilized to improve the potential App Recommender System. This industry objective is based on the premise that such a product will improve the consumption of Apps by the users through better understanding of their usage behavior and added level of personalization thereby potentially increasing the business capacity of such systems. It will also provide opportunity to apply such reasoning techniques and the theories of context awareness in other similar applications and products and reusability of components being developed in parts thereof or in totality in different systems.

Scientific Goal

From a scientific point of view this thesis is expected to contribute to an enhanced understanding of how to apply and extend established design theory and design methodology in order to improve the capabilities in a context aware recommender system. An important element in achieving this will be designing a generic flexible framework that can be applied in any future contextual reasoning based recommender system. Other important aspects are to analyze and define different reasoning mechanism involved, proposing algorithms to achieve them, critically evaluate the different existing methods and identifying suitable ones and the need to be able to define the system independent of the technology domain of the design solutions.

Problem Formmulation

The ubiquitous nature of mobile devices provides added sources of information that, if effectively utilized, can help address these challenges and add value to traditional recommender systems. A major draw of the thesis has been in identifying ways to utilize these information to achieve higher level of personalization.

As we can see from the mentioned works above in section 2.3 and 2.4 that the current systems suffers from one of these problems

- They either need data for cold start training
- They are targeted to specific domain
- They require the user to train the system proactively
- The system is not flexible enough to add new sensors, contexts and situations seamlessly without affecting other components
- They lack the capability to explain the recommendations to the user in a better way and provide the opportunity to learn about the users.

For this to be dazed a framework generalized to be applied to any such systems should be designed and be applied. The designed framework should be used for any context aware recommender system irrespective of the domain of the application. The various components of the framework can have varied implementation based on the system requirements.

The concept of contextual reasoning around the sensor originated data from the android OS was the one chosen for further research and implementation. In our approach we will try to model the relationship between context and situation effectively so that the application can map the sensor originated context data to predict the situation of the user with some degree of certainty. The outcome of the contextual reasoning will add an extra level of information filtering achieved on the User data to recommend him with apps.

The design requirements of the system were that it can handle cold start issues, doesn’t requires the user to train the system proactively and that is flexible enough to add new sensors, contexts and situations seamlessly without affecting other components and is capable to provide good enough explanations of recommendation with capacity to learn misconstrued interpretations.

Concept

A sensor-context-situation-recommendation flow was proposed during the thesis to model the sensor originated data into contexts and use contexts to define a situation as described in figure below

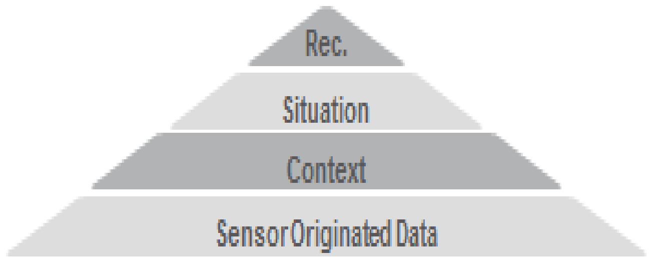


Figure1 – Sensor–Context –Situation-Recommendation Diagram

It is important to define here what are contexts and situations that we are referring to. A situation can be defined as a position or status with respect to the conditions or circumstances the user is in. A situation in our case is same as what the users perceive their state as. A situation consists of different characteristics each of which we call as a context. A context can be determined by reasoning around one or more of the sensor readings. A situation is generally described by a few contexts together with an expected value of each of them. A bottom up approach is more suitable as the situations as a concept of human understanding is too broad and too complex to be modelled perfectly, A bottom up approach will illustrate a system where situational awareness can be considered when making e.g. App recommendations. Situation as an input to recommender systems is important because users in different situations tend to have different consumption habit.

In a classical collaborative filtering method the users are provided with recommendations based on what other users similar to them have procured. While the user may be interested in items procured by similar users he/she may be more interested in items procured by similar users in a similar situation. The basic concept of the system is to provide mobile device users with personalized recommendations of mobile applications by using collaborative filtering and situational knowledge about the user. Each of the situations can be defined by one or more context variables. The context variables are defined by one or more sensor readings and the sensor data originates directly from the mobile device. E.g. a PLACE context could be identified as being “HOME” by analyzing the location history data collected from the GPS sensor. A situation “FRIENDS-FAMILY GATHERING” can be reasoned around by combining the context “SOCIAL RELATIONSHIP”, the context “IN THE SAME ROOM AS” with other contexts such as “NOISE LEVEL”. The predicted situation is then provided as an input to the recommender system.

Use Cases

User sitting with friends

Description:

- Harry is in his home.
- Sitting and chatting with some friends who have visited him.
- Requests for some recommendations

Intended Action:

- The recommender should recommend him with some apps consumed by different users in a similar situation.

User works at a shopping mall

Description:

- Harry works in a shopping mall.
- He is in his workplace and working.
- Requests for some recommendations.
- Due to his surroundings the recommender might think Harry is shopping.
- Recommends him based on consumptions of other users in a situation similar to shopping.
- Harry disagrees and provides the system a feedback that he is working

Intended Action:

- The recommender should learn when the surroundings are similar Harry is actually working and not shopping.

User exhausted after working but still in office

Description:

- Harry is in his workplace.
- Has been working for last 5 Hours continuously and is very exhausted.
- Wants to calm and sooth his mind.
- Requests for some recommendations.
- The recommender recommends him based on consumptions of other users in a situation similar to working.

Intended Action:

- The recommender is correct in this case in judging the situation however is incorrect in judging the needs of the user. To handle this the recommender should diversify recommendations as much as possible to capture users entire taste spectrum.

The Framework

The designed framework can be used for any situational reasoning based recommender system irrespective of the domain of the application. The various components of the framework can have varied implementation based on the system requirements. The framework consists of 4 layers as shown in figure and discussed below.

- Data collection layer
- Context Inference layer
- Situation Reasoning Layer
- Recommender System Layer

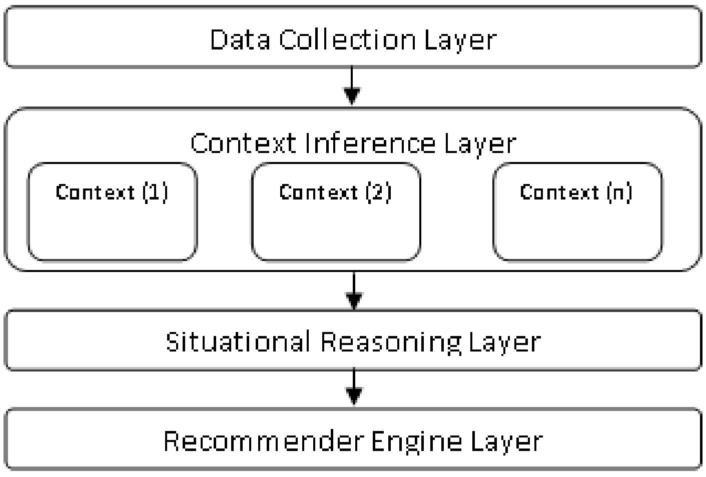


Figure 2 : The Framework

Data Collection Layer

The Data Collection Layer resides in the client application being installed in the device which the users will use. A device may be a mobile device, a PDA or any other dedicated device etc. which possibly has source and access to some sensors. The layer collects the sensor data from the mobile device and sends it to the next layer residing in the server.

Context Inference Layer

The Context Inference Layer consists of independent sensor inference components. The task of each component is to infer the value of a context variable in the system. In identifying the components and their design, a loose coupling approach should preferably be adopted. Each component should be kept as independent as possible. In this way the scalability of the system can be maximized.

Situation Reasoning Layer

The Situation Reasoning Layer consists of a reasoning engine that infers a situation from the context variables. An inferred context can have different types of values associated and possibly even have null values or unhandled values due to lack of available data or unhandled data and therefore the reasoning mechanism should be flexible to accommodate such cases. Since the situations as a concept are difficult to be modeled our study in this domain showed that case based reasoning systems are best suited for this layer over other techniques like rule based systems or Bayesian learning based system. The advantages are further described in section…….

Recommendation Process Flow Diagram

A process flow diagram was constructed to guide the process of development during the Thesis. The diagram incorporates both the existing components and the new extensions to be made to the recommender system which is shown in the below diagram. Due to constraint of time it is not possible to develop all the components in the scope of the thesis and the unimplemented components are mentioned more in the future work sections.

Data Collection Sub-Process

Mobile devices were chosen as the preferred platform for making recommendations. Presence of sensors such as Accelerometer, Camera, GPS, Proximity, Microphone, Gyroscope, Magnetometer etc. provide a lot of input to understand context better in today's Smartphones. It also provides a possibility to reason around the understood contexts thereby adding new dimensions to the personalization. However all this sensors also present challenges such as inconsistent readings, privacy concerns etc.

The device will host a recommendation client application. The application will majorly focus on three tasks namely

- Requesting for new recommendations when requested by the users.
- Collecting application consumption data of the user along with the required sensor data.
- Collecting optional feedback about the user to learn about users situation.

Context – Inference Sub-Process

The context inference sub-process comprises of different components which are loosely coupled from each other, each of which tries to infer a specific context of interest. It receives input when any of the three tasks being performed by the data collection sub-process are called by the user or executed by the system. The input may consist of a stream of consumptions made by the user in a certain interval of time, in which case each of the consumption along with the respective sensor reading at the moment of consumption.

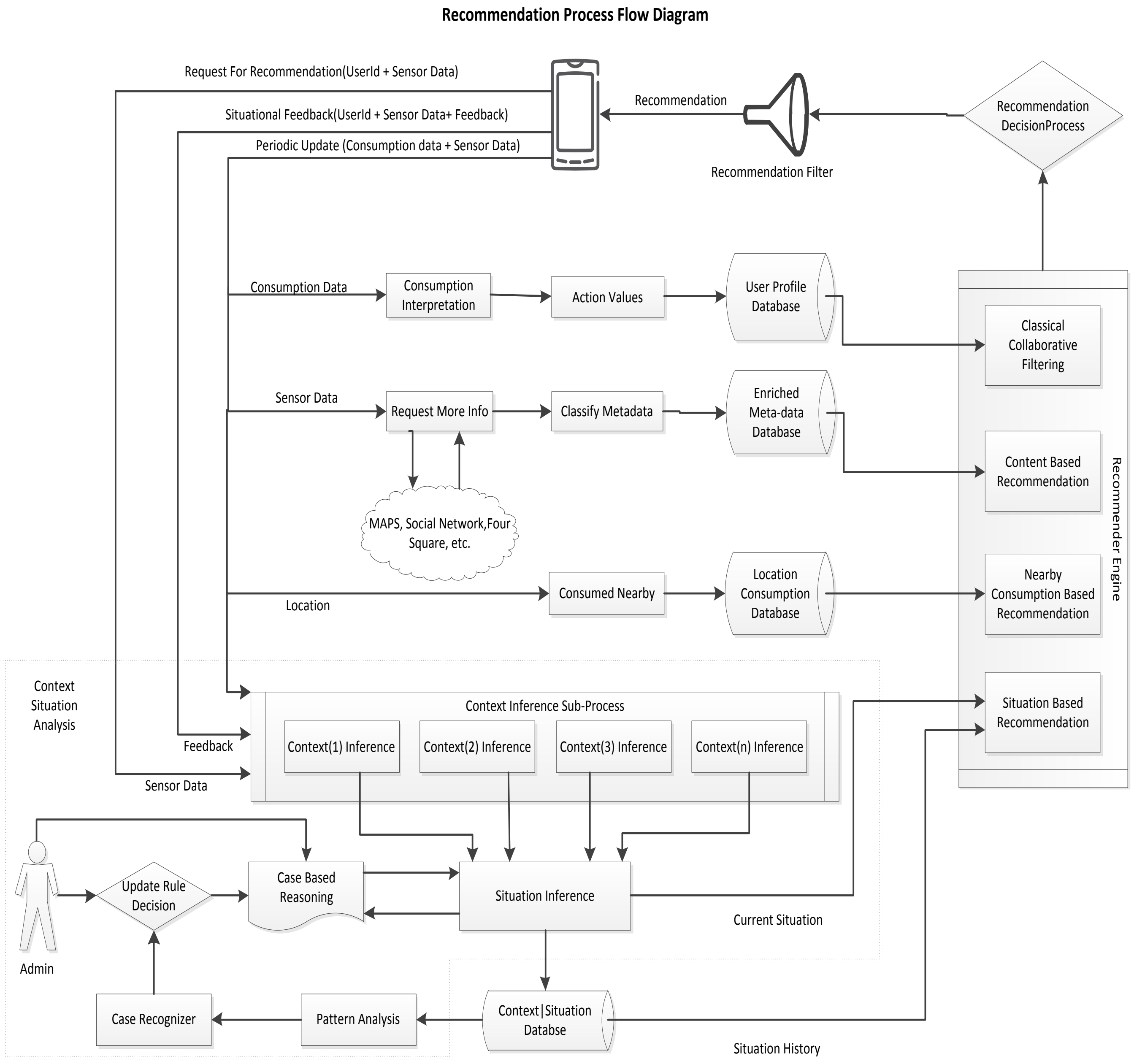

Recommendation Process Flow Diagram
Request For Recommendation(UserId + Sensor Data)
Situational Feedback(UserId + Sensor Data+ Feedback)
Periodic Update (Consumption data + Sensor Data)
Recommendation
Recommendation Filter
Recommendation DecisionProcess
Consumption Data
Consumption Interpretation
Action Values
User Profile Database
Sensor Data
Request More Info
Classify Metadata
Enriched Meta-data Database
MAPS, Social Network,Four Square, etc.
Location
Consumed Nearby
Location Consumption Database
Classical Collaborative Filtering
Content Based Recommendation
Nearby Consumption Based Recommendation
Situation Based Recommendation
Recommender Engine
Context Situation Analysis
Context Inference Sub-Process
Feedback
Sensor Data
Context(1) Inference
Context(2) Inference
Context(3) Inference
Context(n) Inference
Admin
Update Rule Decision
Case Based Reasoning
Situation Inference
Current Situation
Case Recognizer
Pattern Analysis
Context|Situation Databse
Situation History

Figure 3: Recommendation Process Flow Diagram

Each of the consumption is then treated separately by the sub-process and executed independently. In case of request for recommendation and feedback requests, the data consists of one tuple of sensor data. An output of this sub-process consists of inferred values of each of the contexts. The inter-process communication between the context-inference sub process and the client application is achieved using REST APIs and JSON is used to send the data from the client to the server.

Situation Inference Sub-Process

Situation inference sub-process is responsible for inferring the situation of the user using the input from the context-inference sub process. The reasoning mechanism involved is described in the future section. The output of the process is a situation the user is in for a specific set of context values. In case of the request being sent to context-inference is a stream of consumptions the context-inference sends the tuple of context values individually and the situation inference is called for each of the tuples.

Request More Info Sub-Process

This sub-process is responsible for generating meta-data of the apps so that content based recommendations can be generated. Each of the consumptions is marked with necessary info being taken with the help of data that is generated from the internet using the context of consumptions.

Recommendation Engine Sub-Process

The recommendation Engine sub-process contains implementation of recommender engines using various implementations of different techniques like Situation Based Recommendations, Location Based recommendations, Collaborative filtering based recommendations and content based recommendation. Location based recommendation filters the consumption by user's current location and collaborative filtering is implemented using some state of art techniques in collaborative filtering including Support Vector Machines (for recommending items based on content similarities), Alternating Least Squares (a matrix factorization method used in collaborative filtering) and Feature Vector Similarities which were previously developed for generalized recommendation purposes. Each of the recommendation process are called separately either through a call to their interfaces or using REST APIs to get a set of recommendations from them.

Recommendation Decision sub-process and Recommendation filtering

The recommendation decision sub-process aids in combining output from each of the recommendation processes using a suitable weight and merge method and prioritizes the recommendations to create a output list of Apps. The output list is subjected to filtering as per the need of the user and requested by them. A filter can be of two types, Internal filters to make recommendations effective and external filters which choices are given to the user to filter out recommendations.

Situation Learning Sub-Process

It is the sub-process which is responsible for learning about the user's perceived situations which are misconstrued by the system. The mode of learning can be either based on the feedback from the user or through mining data about user's contexts and sensory data to discover new situations that are missing in the system.

Implementation

In this section the implementation details including theory and algorithms behind them for each of the specific components. The emphasis of the implementation has been towards the context awareness framework being built into the system during the thesis work. Figure 4 shows the architecture of situation recommender based on the context reasoning framework presented in section 3.3. Each of the dotted boxes represents each layer in the framework.

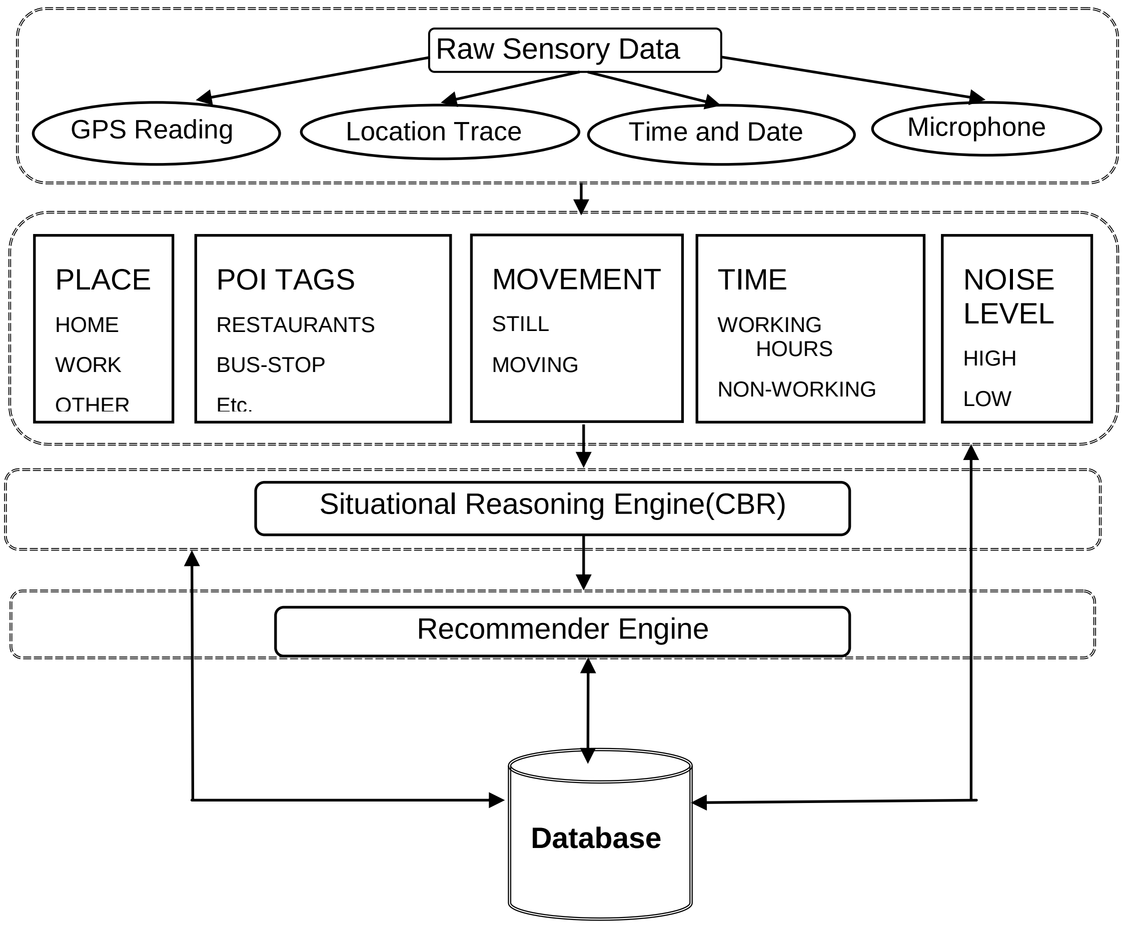


Figure 4: Situation Recommender Architecture

Place as a Context

What places can be of interest and can be inferred?

Place is one of the important contexts which can be used to describe different situations. Place as a context refers to any information that can be generated out raw location data of the user. Places like Home or Workplace can be of immense interest as they can aid describing the most obvious situations when a user is at those places. The location data when aggregated over a period of time can be clustered to infer places like Home or Workplace of the user by plotting against time. Apart from that information about the location can also be fetched from the tags for the location in Maps which is further described in section 4.2.

Algorithm for inferring places

Inferring places from location data is a problem which relates to clustering the raw location data. Due to the ubiquitous nature of mobile devices they are used anywhere and everywhere. Hence to find a location cluster in the data it is not important to partition the data into few clusters rather to find a set of points which are located in close proximity to each other in the dataset. All standard algorithms of clustering were found to be infeasible to cluster data such dataset and hence an algorithm was devised to find the clusters in such dataset which is as follows:

Input: Dataset of Location Co-ordinates with Time.

Output: A set of probable clusters for Location Types.

Step 1: Treat each of the points in the dataset as a separate cluster.

- Maintain an array for count of cluster points. Initialize count to 1 for each element.
- Maintain an array of visited and initialize it to false for each points.

Step 2: foreach Point Pi ∈ P(i1,i2,…in)

- Calculate dij $= R \times 2.a\tan2(\sqrt{a}, \sqrt{(1-a)})$ where $i \neq j$ and visted[j] = false and visited[i] = false
  where $a = \sin^2(\Delta lat/2) + \cos(lati).\cos(latj).\sin^2(\Delta long/2)$

- If(dij < dthreshold)
  - lati = (lati*count[i] + lat2*count[j])/(count[i]+count[j]);
  - set visited[j] = false
  - set count[i] = count[i] + count[j]

```
Step 3: foreach point Pi∈ P(i1,i2,…im)

    • If(count[i]/n >Confidencethreshold)
    • Add Pi to ClusterAnalysisResult

Step 4: Return ClusterAnalysisResult
```

Optimizations

Since after the cluster analysis has been performed it is important to infer what those clusters are, one has to take into consideration the time of being in that location so as to infer places like Home or Workplace.

In the system the clustering is performed from the gathered data of user's consumptions. Since while storing the consumptions the context of time is computed it is reused in order to avoid redundant computation.

From the consumption data where context time is non-working hour the clustering is performed and the highest confidence cluster is used by the system as Home location of the user.

Similarly from the consumption data where context time is working hour the clustering is performed and the highest confidence cluster is used by the system as Workplace location of the user.

Threshold on number of records required in order to perform clustering needs to be set for effective computation.

Since the process is computationally intensive process the computed user location is cached in the database in order to prevent re-computation before a specified amount of time. However, the models need to be updated after a specified amount of time taking into consideration the new set of data of the user to update any change of the behavior of the user. In the current system the time to update is set to 2 days.

POI Tags as a Context

POI Tags for Context Inference

Historical data can be used to find limited contexts from the location data. However there is other location data that is available in the internet like in Maps and GIS applications like Google Maps, Open Street Maps etc.

This context is identified using the available data around the user's current location by fetching tags from maps, GIS applications, Foursquare etc. In our implementation we used tags from Open Street Maps in a 200m radius from the current location of the user. The tags were then assigned a score by calculating the proximity of the tag to the current location.

The tags are then classified by tag types according to the following schema to certain defined classes which to some extent could have an impact on the usage. The classes with higher scores provide the value of this context with a likelihood of being in. A more detailed schema of open street maps is mentioned in Wiki of open street maps.

Going To Travel →	
	"aerodrome"
	"aeroway"
	"terminal"
	"airport"
	"railway_station"
Eating Out →	
	‟restaurant"
	"fast_food"
	"food_court"
	"cafe"
	"ice_cream"
Bank →	
	"atm"
	"bank"
	"bureau_de_change"
Leisure →	
	"leisure"
Healthcare →	
	"pharmacy"
	"hospital"
	"dentist"
	"doctors"
Entertainment →	
	"arts_centre"
	"cinema"
	"social_centre"
	"studio"
	"theatre"
Shopping →	
	"shop"

POI Tags for Meta data enrichment

In the previous work by C. Davidsson and S. Moritz [15] it was established that mobile apps severely lack available meta-data, which made it hard to perform content based recommendations on them.

A positive outcome of our design was that it gave us the opportunity to address the above problem by developing an approach to generate more metadata for apps using the contextual information available from various other sources such as Foursquare, GIS services and maps. An example of this is when a user uses an application at, say a restaurant. It may or may not be related with him/her being in the restaurant. However, when large number of users uses a specific app in restaurants, it may be useful associate the word restaurant with the App. In this way the tags can be used in a collaborative approach to generate suitable metadata content for the apps.

Assume an App which is consumed in two different locations. From the maps and GIS services the system fetches tags for the App. However, there may be some tags which are common in both the sets. If the scores of these tags are added in the case they are common then eventually in a period of time the related tag will outscore the unrelated tags in the system and the high scoring tags can be viewed as metadata of the App. A score here can be for example the based on the proximity of the tag to the current location. Figure 5 demonstrates a simple outcome of such an approach used. The arrows represent consumption and the length of the arrow signifies the score. When the scores for the consumption are added based on the tag eventually over a period of time one of the tag outscores other as in this case the tag supermarket does. Similarly user data from social networks etc. can also be used to enrich the metadata.

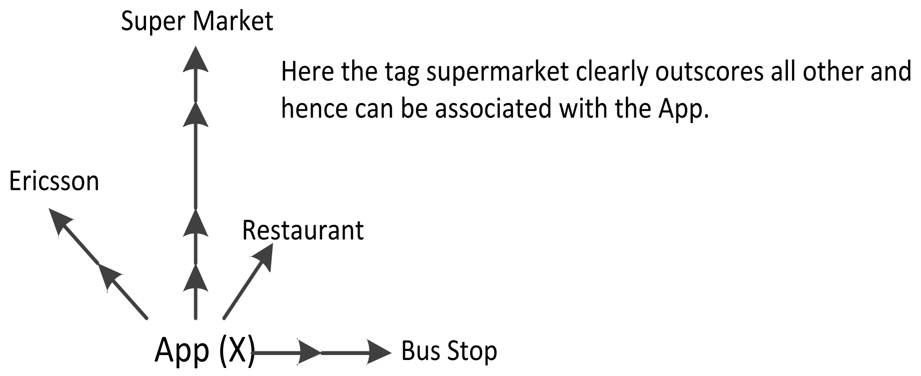

Super Market
Here the tag supermarket clearly outscores all other and hence can be associated with the App.
Ericsson
Restaurant
App (X)
Bus Stop

Figure 5: Outcome of tag association for metadata enrichment

Movement as a Context

While one GPS reading results in a single location at a certain point of time, a series of such readings collected over a period of time results in a trace of location. A background service was implemented to collect such a trace from the device to provide data for movement as a context. Once a trace is received it is not sufficient to infer the context directly by comparing two consecutive points in the space due to inconsistencies in readings of the service but rather a detailed analysis is required as described by [18]. One has to take into consideration to discriminate an observed point into one of two categories: moving point or staying point. If there is no observation error, this identification process is not necessary. However, the observed values of two coordinates at that time may not always remain at the same point even if a traveller stays at one point. This is caused by the noise in the data collection system such as the fluctuation of signal strength. The determination process of the location position of a mobile object also produces measurement errors. An algorithm was formulated based on the Move or Stay algorithm in [*] which is described below.

Algorithm for movement analysis

```
Input : Trace(P1{lat1,lon1}, P2{lat2,lon2},….., Pn{latn,lonn})

Output : Boolean(IS_MOVING)

Initialize : State ← staying, Nstaying←0, movement_value ← 0

Step 1: For each point Pi ∈ (P1,P2…Pn)

  a.  CalculateDistance($P_i$,$P_{i+1}$)

  b.  Switch(state)

      i.  Case: staying

          a)  If(distance > $d_{threshold}$)
                  State← moving
                  $N_{staying}$ ←0
                  movement_value ← movement_value + 0.5
                  (traet point as provisionally moving)

          b)  else

                  $N_{staying}$ ← $N_{staying}$ + 1
                  $Lat(i) = \frac{1}{\text{Nstaying}} \sum_{i=t,\text{Nstaying}} Lat$
                  $Lon(i) = \frac{1}{\text{Nstaying}} \sum_{i=t,\text{Nstaying}} Lon$

      ii  Case :  moving
            If(distance> $d_{threshold}$ )
                  movement_value ← movement_value + 1.0
```

```
        else
            $N_{staying} \leftarrow 2$
            State $\leftarrow$ staying
                $Lat(i) = \frac{1}{Nstaying} \sum_{i=t,Nstaying} Lat$
            $Lon(i) = \frac{1}{Nstaying} \sum_{i=t,Nstaying} Lon$

Step 2:
  if(movement_value >THRESHOLD_MOVEMENT)
          return true
  else
          return false
  Procedure CalculateDistance(Pi,Pi+1)
          $a = \sin^2(\Delta lat/2) + \cos(lat_i).\cos(lat_{i+1}).\sin^2(\Delta long/2)$
          $d_{i,i+1} = R \times 2.a\tan2(\sqrt{a}, \sqrt{(1-a)})$
          return $d_{i,i+1}$
```

Explanation:

The algorithm marks each of the point in the trace as in 3 states (a) Moving, (b) Provisionally Moving and (c) Staying, based on the states of the preceding points. To achieve this it maintains states of the preceding points till the state of the point changes. If the point is staying at time t and the distance of the next point is more than the threshold distance then the algorithm marks the point as provisionally moving, if the point next to a provisionally moving point is moving them the algorithms marks the point as moving. For each consecutive moving point the algorithm updates the average of the latitude and longitude of those points.

For each of the provisionally moving point it updates the value of movement_value by adding 0.5 and for each moving point movement_value is added by 1.

In the end if the movement_value crosses a certain threshold then the trace is analyzed as moving else false.

Date and time as a Context

Time as a context can be utilized in different ways depending on the requirement of the model to be constructed. E.g.

- Day(Monday, Tuesday,…………..)
- Date of the Month(1,2,3,….31)
- Is Working Hour(Y/N) - Person dependent – (Simple assumption 08-19 / 19-08)
- IsHoliday(Yes/No) – Complex country dependent – (Simple assumption Saturday/Sunday/common Holidays)
- Hour(Morning (5-10), Noon (10 – 13), AfterNoon(13-16), Evening(16- 20), Night(20-05)

In our implementation Date is used as an input to compute if it is a holiday or working day of the user. A holiday database published by Outlook was used for this purpose. The database has been populated with holidays of all the countries till 2020 after which the same will be needed to be updated. Time is used to compute if it was a working or non-working hour. Charging behavior is also a good indicator for working hours of a person as [19] shows that almost all working person charges their phones between 1800 hrs. to 0500 hrs. To simplify things we have currently assumed that working hour of a user is from 0800 hrs. to 1800 hrs. in a working day and the remaining hours are non-working hours. This is similar to creating date time profiles for example in [11]. More detailed usage pattern can then also be studied by correlating e.g. Wi-Fi network type, phone charging behavior etc. to working or non-working hours.

Noise level as a context

Microphone reading can be a good indicator for different situations a user might be in. Eg. When a high noise is coming from the microphone and the user is in home then probably he is watching TV or listening to music. The same voice sample can be analyzed against a set of samples to determine the exact kind of the sound. However recording sound is a matter of much higher privacy concern than say getting location or accelerometer readings which needs to be accounted for. The instantaneous microphone reading is collected by the data collection unit when a recommendation is requested. The maximum amplitude of the wave gives an indication of the amount of surrounding noise which is sent back to the context analysis layer. Currently the system tries to analyze the noise level by comparing it with some threshold values which indicate the context values. However, one could also apply sound analysis to figure out which type of noise it is. Such as, sound from a TV, sound from an CD, sound from traffic etc.

Situation Reasoning Engine

An introduction to case based reasoning

A case is “a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to archiving the goal of the reasoner” [Kolodner, 1993].

A case based reasoning system has its knowledge and expertise embodied in library of past cases. Each case ideally contains a problem definition along with a solution or outcome to the problem. The knowledge or the steps required by an expert to solve the problem is not recorded in the solution but is implicit to the solution.

A case based reasoning (CBR) system works very much like the human mind when it faces a new problem. When we face a new problem our mind generally thinks of how we solved the problem last time we encountered it or our mind tries to figure out did we faced similar problem in the past and if so how did we solved it. This is especially true in a child's mind which is the reason they learn things so fast.

A case based reasoning system works in the same way. To solve a current problem it tries to match the current problem against the cases in the case base and retrieves a set of similar cases. The retrieved cases are used to suggest a solution of the current problem to the user. The suggested solution may be tested and evaluated for success and reused. If the solution is then revised the current problem and the solution are held in the case base as a new case.

In a CBR system a case is described by a set of features each having a value in defining the feature. A feature can be of any data type supported by the CBR System. Each of the cases are compared to each other statistically by the system to compute the similarity between them.

A typical lifecycle of CBR System is shown in figure 6:

At first instance a CBR system may look similar to a rule based reasoning system, however there are fundamental differences between a CBR and a Rule based system.

A rule based system requires eliciting an explicit model of the problem domain. As we all know the process of knowledge acquisition has a lot of other associated problems. In contrast a CBR doesn't require any explicit model. Cases that identify significant features are gathered and added to the case base during development and remaining can be added incrementally after the deployment is over. Thus it is easier to develop case base without passing the knowledge acquisition bottleneck.

A CBR system is much faster and easier to be implemented than constructing a rule-based system. Case bases do not have to be complete when they are deployed for use, as even non-computer experts can add cases to the existing structure or the cases can be added with the help of feedbacks from proposed solutions or objective evaluations.

The output of a CBR system can be multiple cases with a hit probability of each case while the output of a rule based system is ideally a single solution or no solution. Case based systems are easier to maintain and adding or deleting cases is much easier than similar operations in rule based system.

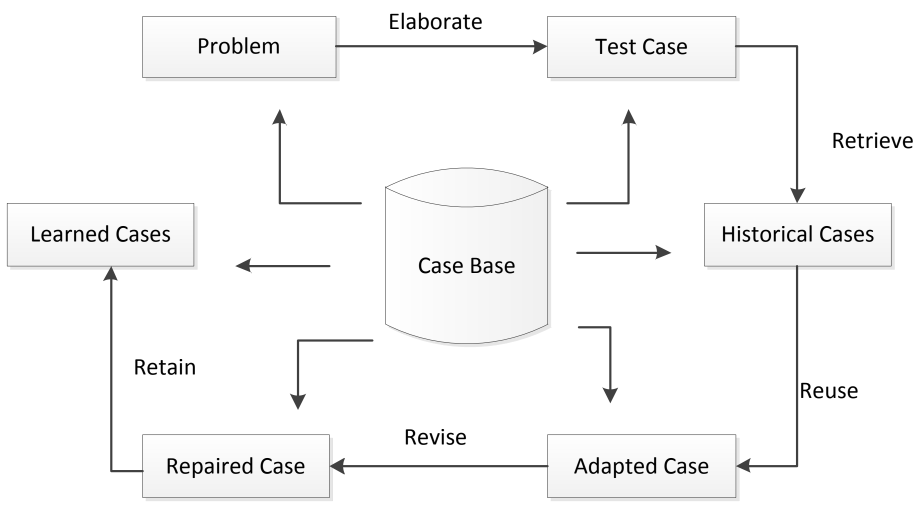


Figure 6: Lifecycle of a CBR

Why case based reasoning for situation analysis?

Situation as a concept of human understanding is too broad and too complex to be modelled perfectly. The situational reasoning is called after the context analysis is performed. It may be possible that some of the context may remain un-inferred by the system due to lack of available data or unhandled input data. E.g. POI tags around a location may be blank if no points are marked in the maps in a location near the user. The reasoning engine should therefore be flexible to accommodate such cases.

Situations for two users having the same context values can be different and thus is not absolute but relative entities in modelling them. The cases can be very close to each other and in such scenario the system can leverage by trying to propose to the user the best possible solutions and thereby learning from acceptability of them by the user.

A CBR system is found to be best suited for this, as the problem of situation modelling is not well defined, completely understood and varies from person to person.

A learning module is integrated by collecting optional feedback from the user about the perceived situation and thereby adding new cases to the case base on a new situation. Remembering new cases helps to avoid past mistakes. Also CBR systems provide means to evaluate different potential solution which further can be used in ranking, ordering and filtering of the recommendations. In comparison to a rule based system adding new cases are often quite easier in a CBR system. A brief flow diagram is shown below in Figure 7.

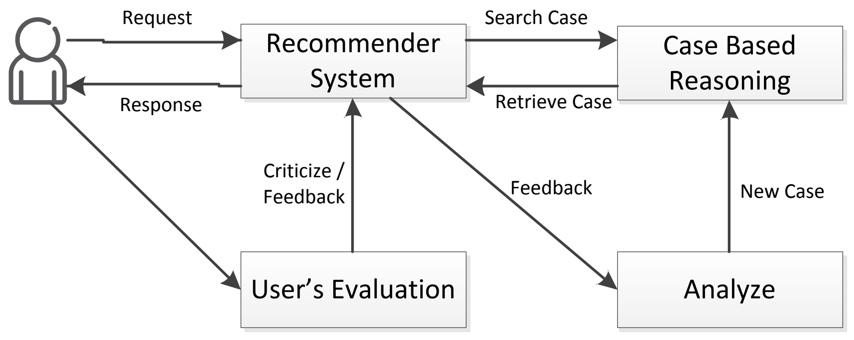


Figure 7 : Flow diagram of interaction between Recommender System and CBR

Implementation details of the CBR.

The implementation of the case based reasoning has mainly 5 major tasks which are listed as follows and discussed below:

- Choosing and creating data source for storing cases.
- Constructing the input query.
- Retrieving records.
- Evaluation of retrieved records.
- Learning new cases.

<u>Choosing and creating data source for storing cases.</u>

The possible data sources for a CBR can be Databases, XML files, Spread Sheets etc. which can be used as per need in various scenarios. Spreadsheets are generally used by CBR when the intended administrator of the system is non-technical persons like managers maintaining an organizational case based reasoning system. In this case as there was an already implemented database for the system present, the data source was chosen to be the same data base by adding separate tables for the CBR system. Each distinct feature which constitutes the problem space for all the cases in the system is stored in the features table. Each feature in this system actually represents a context or a part thereof. The various cases are stored in the case table and the output i.e. situations are stored in the situations table.

<u>Constructing the input query</u>

This is the task of mapping the problem to the features and creating a new case to be investigated for solution. In this system this problem refers to mapping the context values to the set of features and then creating a new case to be searched in the system. A static mapping is used currently in the system which can be configured to a dynamic mapping in the future.

Retrieval of Records

Retrieval of records is the process by which the retrieval algorithm finds the nearest matching case from the case base to the current problem in hand. Case retrieval process is a combination of search and matching algorithm. Two of the widely used techniques for case retrieval in a CBR system are:

- Nearest neighbor retrieval approach
- Inductive retrieval approach

Nearest neighbor approach:

In a nearest neighbor approach the system tries to compute the similarity between the stored cases and the new input case based on the values of the associated features and their respective weights using some evaluation function like

$$Similarity\ (Case_I, Case_R) = \frac{\sum_{i=1}^{n} w_i * Similarity(f_I^i, f_R^i)}{\sum_{i=1}^{n} w_i}$$

Inductive retrieval approach

Inductive retrieval algorithm is a technique that determines which features do the best job in discriminating cases and generates a decision tree type structure to organize the cases in memory [17]

Situation based Recommender System

How to recommend items based on situation?

In our implementation of the recommender system we are focusing on providing the users with recommendation of the apps consumed by similar user in a similar situation. While in classical user based collaborative filtering a user is provided with recommendations of the items the user's similar to the given user has consumed, these items falls broadly into two categories if we take the situation into consideration (1) Items consumed by the similar users in same situation (2) Items consumed by similar users in different situation. There is a third category which is Items consumed by dissimilar users in the same situation. We believe (1) outscores the other two and the remaining needs to be differentiated statistically or using heuristics.

There are two ways in which this can be achieved:

- *Applying a three dimensional collaborative filtering **R: Users x Items x Situation → Ratings.** This is similar to doing context collaborative filtering. There are many advanced algorithms available on three dimensional collaborative filtering. It is very useful to do it this way if the collaborative filtering system is not already implemented.*

- *If the collaborative filtering algorithm is already implemented in two dimensions i.e.* ***R: Users x Items → Ratings*** *as in our case the same can be extended by combining the output of the collaborative filtering with that of applying filter on the consumption data based on the situation, the steps of which are described below*

```
Step 1: Situation Recs ← Get consumption where Situation = Situation(a) order
by Situation Value, Time of Consumption

Stpe 2: Normalize and Sort Situation Recs

Step 3: Get Collaborative Recs from collaborative filtering.

Step 4: Normalize and Sort Collaborative Recs

Step 5: for each x: Situation Recs

        If Collabolrative Recs contains(x)

                x.weight = x_collaborative.weight + x.weight

                Remove x from collaborative recs

Step 6: Add remaining collaborative recs

Step 7: Normalize and Sort
```

Data collection process

For implementation purpose the Android platform was chosen because of the many advantages. It is a flexible and attractive platform for building real world applications and is maturing rapidly. In this section some of the features of the Android platform to support data collection is listed.

Android OS features to support data collection in phones

The two major tasks of data collection in an App Recommender System is the task of

a) Collecting App consumptions.

b) Collecting Sensor data.

Collecting App consumptions is the process of collecting running apps in the system. The Android application Framework contains of very important component named ApplicationManager the task of which is to manage running applications in the system and hosts many different services one of which is "getRunningAppProcesses". A call to this method returns a list of all running apps in the system in that instant. The process of collection is implemented as a background process which runs periodically at a certain interval of time and collects the running Apps.

Once the running apps at any instance is collected it is important to associate the right context values to the particular consumption and for which different sensor data should be collected by the system to associate with the consumption. The Android platform is ideal for developers to create innovative applications leveraging on the use of hardware sensors available on the mobile phone due to its excellent available interfacing options to the sensor subsystem and media recorder. Some of the interesting hardware oriented features available in the platform includes the following.

- android.hardware.SensorManager
- android.hardware.SensorListener
- android.location.LocationProvider
- android.media.MediaRecorder
- android.hardware.Camera

SensorManager is the class that permits access to the available sensors in the system. The SensorManager contains different constants to represent aspects of Android sensor subsystem including Sensor Type (Accelerometer, Light, Temperature, Orientation, Magnetic Field etc), Sampling Rate (Fastest, game, normal), Accuracy (High, Low, Medium, unreliable etc). The Sensor Manager provides comprehensive interface to access different sensors available in the system. The SensorListener is the interface to be implemented by a class which wants to receive updates to the sensor values as they change in the real time in the system. An application implements this interface to monitor one or more sensor values that change in the system.

LocationProvider is the abstract superclass for the various location providers. A location provider provides periodic update to geographical location of the device. Ideally there are many location providers in the system like GPS provider, Cellular network based provider, internet based providers etc. Each of the location providers have different levels of accuracy. The location package also has different constants to define the access to the providers like ACCURACY_FINE, ACCURACY_COURSE, POWER_HIGH, POWER_LOW etc. which helps in choosing the location provider suitable for the application. The LocationProvider also helps in judging the best location provider in the system according to the required criterias using defined methods in the class.

MediaRecorder is a defined class to record media samples, which is useful in recording audio activity within a specific location. Audio samples than also be utilized in identification purposes or any other pattern recognition steps to identify the sound or to upload the sound to any network location for assessments. MediaRecorder class also provides option to record video. There are different encoding schemes implemented in the media recorder which can directly be used by any application like H263, H264, MPEG4 for video and AMR_NB, Raw AMR, 3GPP etc. for audio. The recording is based on simple state machines. Each of the states it enters has to be preceded by a valid state after the initialization.

Process of Weight and Merge and Filtering

Since the system supported multiple type off recommender engines to achieve diversity in recommendations, it is important to merge the end results of each of the recommenders in a weighted manner to compile the final recommendation list. The system currently supports predefined static weights attached to each of the recommender engines which can hence be made configurable and used accordingly. The output of each of the recommender engines is a list of Apps along with a ranked normalized score of the recommendation in the range of 0 to 1. The scores are then multiplied with the weight of the recommender engine to achieve the final score of the recommendation. The final list consists of all the recommender engines recommendations sorted by the final score of recommendation. The system also currently supports two types of filters to filter out recommendation.

- Filter by recommendation type – In this case the specific recommender engines are called and final list of recommendations are compiled only using their output.

- Filter by category – The output of each recommender engines are filtered for the requested categories and the process of weight and merge is applied back.

Recommendation Explanation & Situational Feedback

Explanations in recommender engines are quintessential in building trust of the user. Since the final output of the recommender engine is actually coming from different recommenders it is necessary to tell the users in a human understandable way as to why they received the recommendation. The system maintains different type of explanation for each of the type of recommendation. E.g. A location based recommendation is given an explanation “This app is recommended to you based on the apps popular around you” while a random recommendation is given an explanation “This app is recommended to you at random”. In case of situation based recommendation it is not just enough to tell the user that the app was recommended based on your situation as that virtually adds no information to the user. Rather we tried to propose the explanation along with the perceived situation of the user by the system.

Due to the insufficient domain knowledge in modeling situations due to its broad and vast concept and the problem of knowledge elicitation situation modeling required feedbacks from the user to learn about the situations for which the case based reasoning approach was chosen. The situational feedback can be of two types

- Implicit situational feedback: Learning the frequently occurring context patterns for the users by frequent pattern mining on the generated cases being queried into the system. This step was proposed during the thesis but not implemented due to constraints of time.

- Explicit situational feedback: Explicitly asking the user is their perceived situation correct. This might be a step which may bother the user if asked every time by the system and hence we implemented the feedback as an optional step that can be filled by the user if they disagree with the perceived situation and want to provide a feedback.

The perceived situation is provided as part of the explanation of the recommendation in case of the recommendations from situation based recommender. Once the feedback is provided the input values at that instant is analyzed and a new case is retained by the system. This creates a learning cycle in the recommender system.

Qualitative Evaluation

A through system testing was performed after the implementation to verify the expected behavior of the system. Modifications were made wherever required and optimizations were applied wherever it permitted. Some of the challenges faced during testing of the system are presented in the discussion section. Some of the Hypothesis which were drawn initially and fall under the purview of the designed system are analyzed and presented in detail in this section. Consumption behavior of the users in different situations was also analyzed during this phase which is also presented in this section along with the utility of different type of recommendations by the user.

Hypothesis Verification

Towards expanding the context and addressing cold start

- ***Hypothesis 1:*** *Data available with the Telephony Manager in the Android OS can be effectively utilized to analyze patterns. Example of such data can be Network Operator Name, Sim Operator Name, Cell Location, Phone Number, whether in roaming etc.*

  The TelephonyManager class in the Android platform provides access to many such information when the proper rights are given to the application by the device owner. These rights can be requested by the application to the user when the application is being installed by the user on the android platform. Few of the information being utilized currently in this implementation are

  a) **Network Country ISO Code**: To identify the country where the user currently is which is in turn used to determine if the day is a holiday in the current country of the user using holiday database.

  b) **IMEI Number:** The MSISDN number is currently used by the system to identify each of the users uniquely by the system. To preserve the privacy of the user the MSISDN number is encrypted using MD5 algorithm in the client side and then sent to the server. The use of MSISDN number enables the system to keep knowledge about the user and their preferences even if the user changes his mobile phone but uses the same mobile number/ SIM card.

  Hence on these demonstrations it can be said that the Hypothesis 1 stand verified

- **Hypothesis 2:** *User Demographics information either collected via the Mobile Operator or collected from the User directly from the recommendation app can be utilized effectively to analyze patterns.*

  After consultation with few experts in the telecom domain it was known that such data can be collected about the user if proper collaboration is made with the operator. However such arrangements can be difficult to obtain and have a lot of complicacies involved. Also it is generally seen that the end users are reluctant to share their personal information due to lack of trust in the system which is the reason why this path was not chosen for a complete verification.

- **Hypothesis 3:** *Content of the metadata of the apps can be effectively utilized to find a list of relating apps and generate the similarity measure of the apps.*

  It was found from a deeper background study that the mobile apps lack metadata severely [15]. To address this problem we presented an approach in this thesis on how to enrich the metadata in section 4.2.2. Hence this Hypothesis stands partially verified.

- **Hypothesis 4:** *Making it optional to provide the demographics data for the user and providing them with statistics of how the data influences the recommendations will reduce the concerns of the user in providing personal information.*

  Collecting demographic information was not a focus area during the thesis due to the mentioned problems in analysis of hypothesis 2 as a result this hypothesis also stands unverified.

- **Hypothesis 5:** *Providing users with explicit feedback options, e.g. Like/Dislike Agree/Disagree buttons can help in better User Modeling.*

  Agrre/Disagrre buttons were provided to collect optional situational feedback from the user which helped the system to learn the situations of the user better and achieve better situational modeling.

Towards expanding to Social Network, blogs etc

- **Hypothesis 6:** *Providing the user with an option to share interesting apps on social networking sites will open new source of generating recommendations based on what apps the user's friends are sharing etc.*

- **Hypothesis 7:** *Unstructured data from the web, blogs etc can be a potential source of metadata of the apps.*

- **Hypothesis 8:** *Semi Structured data on twitter can be utilized effectively to discover relations between apps.*

  Three of the above hypothesis was left out as the scope of these three hypotheses will in itself form a separate topic for a thesis. The direction towards social recommendations is very interesting and has been implemented by lot of established recommender systems in the market and may be very good step in improving the recommendations in the system further.

Towards diversifying the recommendations

- **Hypothesis 9:** *List of recommendations can be diversified based on recommendations from different classes of relating apps.*

  The list of recommendations in the current system are diversified by combining outputs from 4 type of recommender engines to fetch the final recommendation list in the system namely (a) situation based recommendation (b) location based recommender (c) consumption based recommender (d) random recommender. So the hypothesis stands verified by the current system.

- **Hypothesis 10:** *The recommendation engine can adapt to the amount of recommendations that is provided for different classes of recommendations based on the user's actions over a period of time dynamically.*

  This hypothesis was not verified in the current scope of the thesis however the functionality is a logical extension and is a potential future work to be performed on the system and hence the hypothesis stands unverified at this stage

- **Hypothesis 11:** *The list of recommendations can be filtered in multiple levels to achieve diversity.*

  The list of recommendations is filtered in multiple levels in the current system. The system currently supports two filters (a) Filter by Category & (b) Filter by recommendation type. Hence the hypothesis stands verified in the current scope of the thesis. More filters may be added later to the application.

Consumption Analysis in different situations

For testing and verification purposes the developed system was distributed to few users after development which generated some amount of data about the consumption and utilization being done by those users. Although the scope of the thesis didn't included usability tests, however a few tests were performed in the end. The consumption data of different users were analyzed to verify a basic assumption in situation based recommendations that the Users in different situations have different needs.

Since the user base was small and the data were collected for a short period of time only those situations were analyzed for which there were at least some considerable amount of data. The results from of the following 3 situations were compiled and are presented in the pie charts.

<u>Idle at home</u> – The top three categories in the consumptions of apps by the users in this situation had been “Brain and Puzzle”, “Entertainment” & “Social” While they were followed by the categories of “Tools” and “Communication”. So clearly we can say that the consumptions of the user when the system perceived their situation as the this were often towards the apps meant for relaxing and soothing the mind then say of providing the users with productivity.

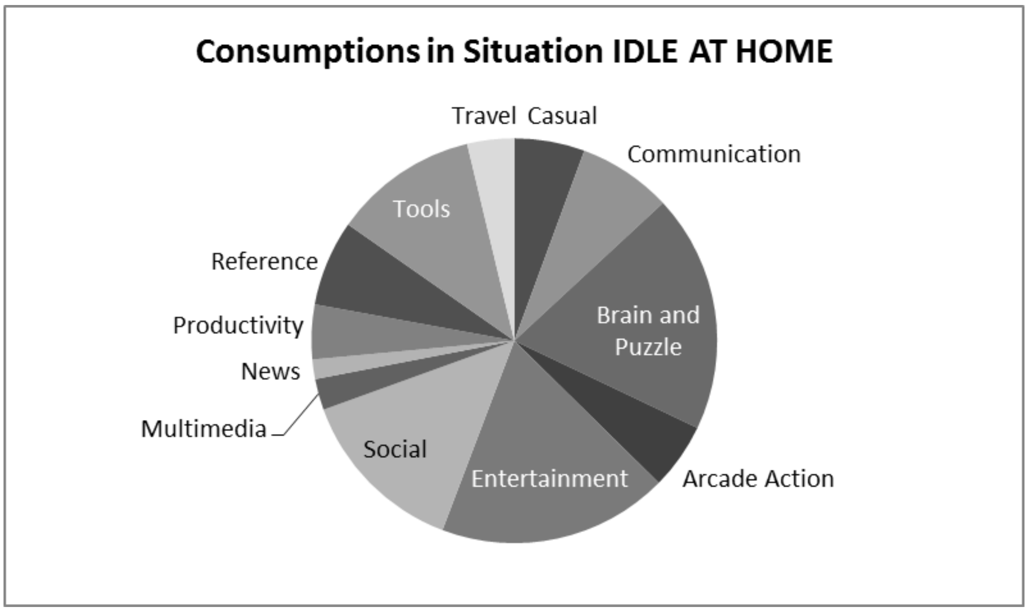


Figure 8: Consumptions in Situation IDLE AT HOME

<u>Working</u> – When the system perceived the situation of the user as working the top three consumptions categories were “Social”, “Tools” and “Communication”. The users here in this case were less interested in “Brain and Puzzle” but were interested in apps relating to “Travel” and “News”. While their interest in apps of the category “Casual” almost remain identical.

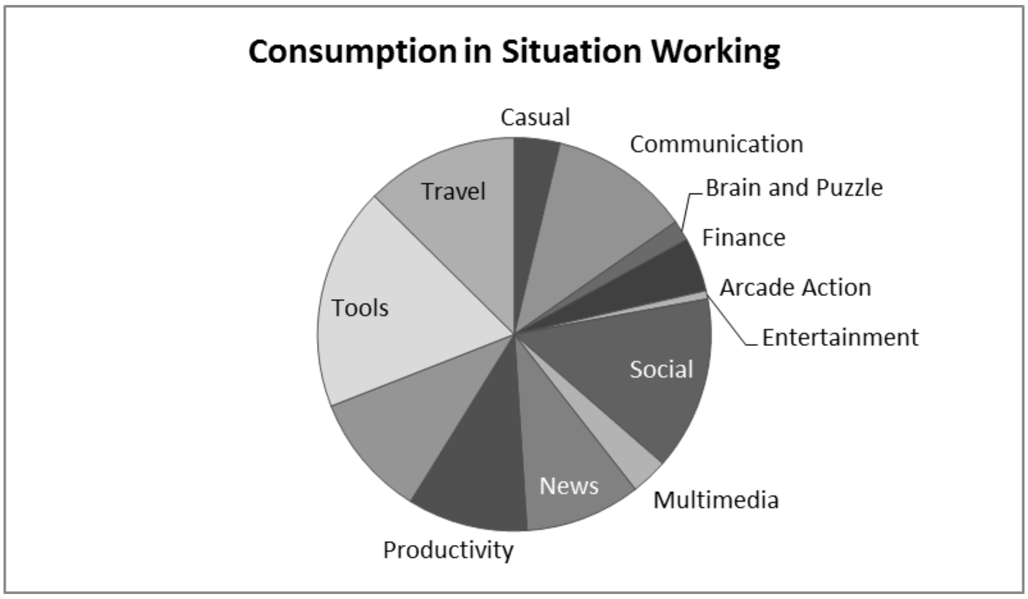


Figure 9: Consumptions in Situation WORKING

<u>Shopping</u> – When the perceived situation of the user by the system was ”Shopping” the top three consumed categories of Apps were “Tools”, “Productivity” and “Travel” which were followed by Apps from the category of “Social” and “Multimedia”.

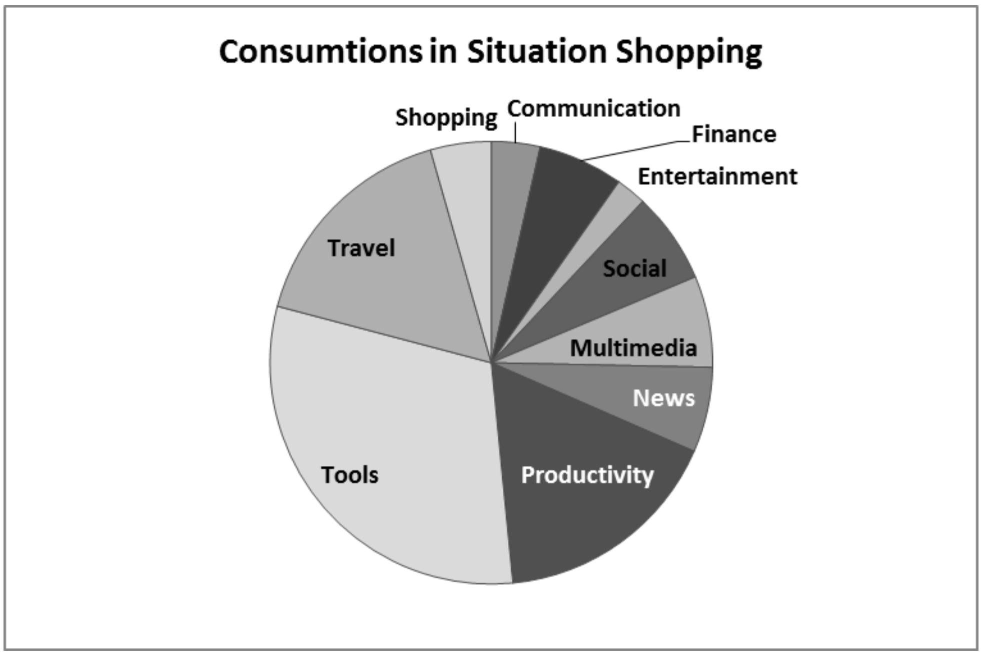


Figure 10: Consumptions in Situation SHOPPING

<u>Summary</u>

It is evident from the above described data that the consumption of the Apps by the users of the system varied in the different perceived situations. While the users in the situation “Idle at home” favored mostly leisure type apps, the users in situation “shopping” and “working” mostly favored productive apps. Some apps of the category “shopping” was also actually logged in the situation “shopping” which was quite encouraging. Some of the unexpected results were many “Travel” category apps being used in the “Working” situation which to a common sense can be said as unexpected however gain the users have varied needs and taste and hence forces the need for diversity in recommendation. The apps of the category “Social” found its usage in almost all situations more or less in high percentage. This behavior is justified as people now a days use Social Network almost anywhere and anytime. The Apps of the category “Games” and “Entertainment” didn’t found much weightage in the situations “shopping” and “working”.

Recommendation Utilization Analysis

The actions of the users were logged whenever they consumed a recommendation to analyze the results. The application shows the recommendations as a list. The elements of the list are the title of the Apps. On clicking the item the user is sent to a next page where the details of the App are available with a button to install the App. By saying “a recommendation is consumed” it is meant that the user has at least propagated from the list to the details page even if they didn’t installed the app. If the user views the details of the App it can be safely assumed that the recommendation has generated some interest in the user. The result in the following chart is compiled by taking into consideration that the user has either viewed the App details or has installed the app and thus represents the recommendation utilization chart.

It is seen that the Situation Recommendations outnumbered the other two. Situation Recommendations comprise of around 46% of the consumed recommendations which throws some light on the need of Situation based recommendations. However the system had a slightly higher weightage to situation based recommendations in the weight and merge process and it will be interesting to study the recommendation utilization by the user in the case where weights are equal or is low for the situation based recommender.

The location based recommendations had the least utilization but that may be due to the fact that the user base was quite low in number and spread around the world and there was very less concentration of the user in a specific location and hence the random recommendations outnumbered the Location based recommendations.

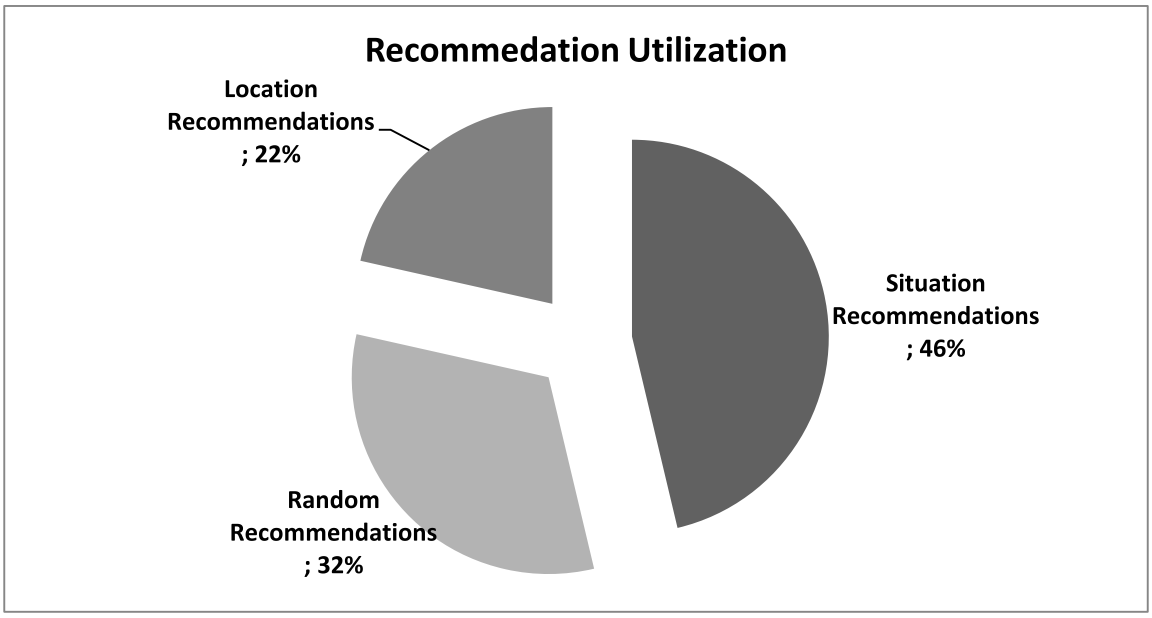


Figure 11: Recommendation Utilization

Discussion and Conclusion

The work presented in this thesis is a first step towards building a situation based mobile recommender system the findings of which show great promise. Although situation as a concept of human understanding is too broad and too complex to be modeled perfectly, our work illustrates a system where situational awareness can be considered when making e.g. App recommendations by utilizing a case based reasoning to overcome the problem of knowledge elicitation.

Situation as an input to recommender systems is important because users in different situations tend to have different consumption habit which was re verified by the data from the implemented system. Defining situations also opens up a possibility to give users better explanations, in a human readable format, of the underlying reasoned context that has been considered when making the recommendation. However, high number of situations in the system may cause ambiguity and affect the response to overcome which advanced retrieval methods must be applied in the case based reasoning system. Data mining techniques, when applied to find frequent patterns of contexts together, can be used to figure out new situations and provide as a way for implicit learning of new situations in the system along with the existing explicit way of learning through the feedback of the user. However, a challenge in such approach will be classifying and labeling the situations which are learned by the patterns of context occurring together. The question that addresses how to find situations that could be relevant to context aware recommender systems that operates in a mobile environment requires a combination of qualitative and quantitative methodology.

Following a layered loosely coupled architecture in situation recommendations eases the process of adding new sensors, contexts and situations. The framework presented in this section 3.4 is very flexible in terms of this respect and so is the implemented system based on the framework.

Having different type of recommenders working in the system helps achieving diversified recommendations. The random recommender helped the system achieve serendipity and increased the recommendation utilization. Diversity in the system is achieved by providing user with the output of different recommenders. Currently if the user requests for recommendation once and doesn’t finds any interesting apps and then requests back in few minutes the system will show the same recommendations if all the state of the user remains identical. However ideally the system should adapt a little based on the first recommendations utilization and try to project some other interesting apps. This may be a good direction to work in the future. However too much personalization causes filter bubble problem in recommender systems and as Eric Schmidt says “It will be hard for people to consume something that is not in some sense tailored for them” may become true.

Another challenge that still remains is the consistency of the sensory data available that effectively could be used for inference. While sensors like GPS, Time, and Microphone etc. are quite consistent if they are permitted to be used and if they are in a usable state while other sensors such as accelerometer etc. are a little inconsistent. On the other hand, the system provides ease in adding new sensors. I believe that in the near future new and more advanced sensors will be available in phones which eventually will help discover the different situations and improve the system immensely.

One of the innovations in the implementation of the system has been the effective use of the POI tags. This which added a lot of meaning to the “Location as context” as mobile users meaning to the “Location as context” as mobile users access their phones from anywhere and everywhere. We have also demonstrated how POI tags can be used to enrich the metadata of mobile apps as a side effect of our system design. This opened up for new opportunities to make content based recommendations which prior had been hard to do on mobile Apps due to lack of Meta data.

As we look towards the future in a networked society, where everything that will gain benefit from being coherently linked will be connected to the network, I believe sensing contexts will have lot more different avenues to enrich any type of service. A flexible system where it is possible to gain knowledge by addressing context-awareness is therefore the need of the hour. The framework is a small step towards this vision and I hope that the situational aware mobile App recommender system and the initial results discussed are motivating and interesting for the community.

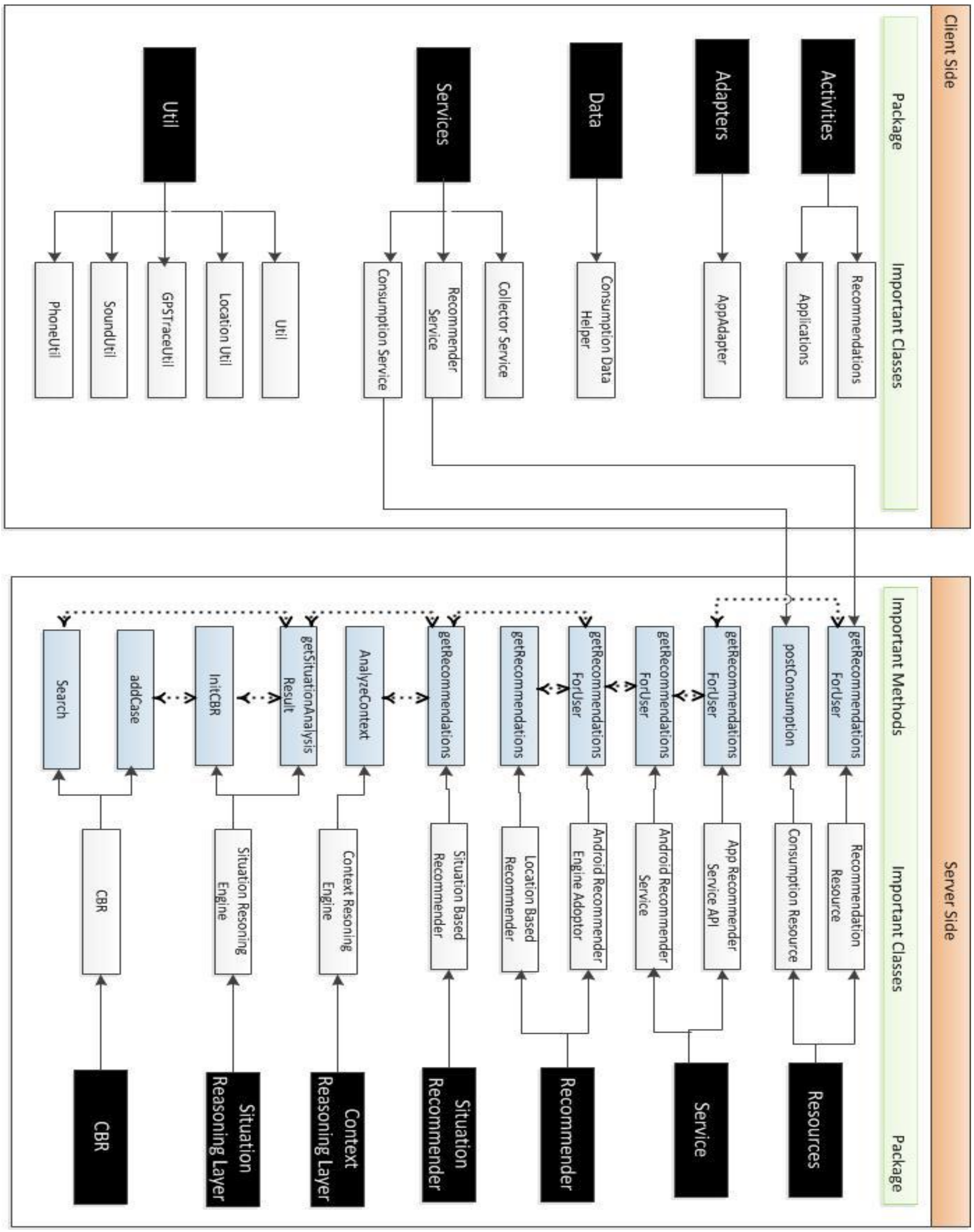
Future Work

Some of the future work that can be performed on the system were listed and presented in this section.

- To add more sensors, contexts and their effective utilization in situation recommendations.
- To implement advanced retrieval methods in the Case Based Reasoning system.
- To make weights of the different recommender dynamic and learn the weights for each user individually from the recommendation utilization data of the user.
- To implement implicit learning of the undefined situations in the system by applying frequent pattern analysis of the context inference results.
- To perform a comprehensive quantitative study of the system.
- Analyze different context reasoning mechanisms and implement new context reasoning mechanisms as identified and deemed proper.
- To integrate system with various other sources of information like GIS systems other than Open Street map, Weather APIs, Foursquare etc.
- To identify performance bottlenecks and apply optimizations in order to transform the system to a production ready state.
- To integrate with social media and effective utilization of data generated therein.
- To use the data shared by the users about apps in social networks in the process of recommendation.
- To extend the search process in the system from searching in the local apps database to searching Apps in the Markets.
- To add diversity in recommendation not just by providing different types of recommendation but also to diversify based on the utilization.
- Adding like/dislike buttons to learn the taste of the user about the apps may be a good idea but might extend the problem of filter bubble and hence can be studied in detail.
- To implement Content based recommendations utilizing the generated metadata.

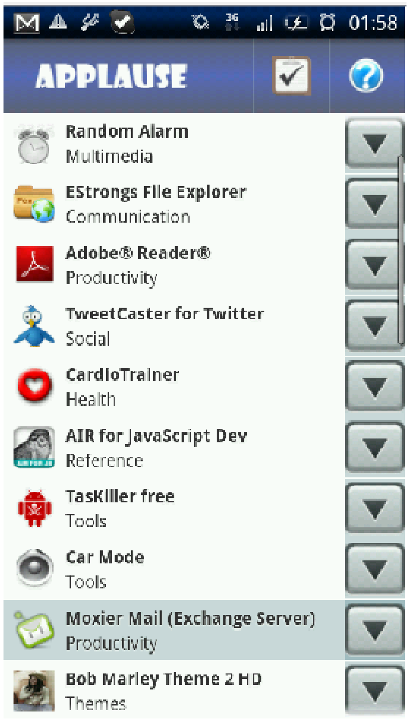
Appendices

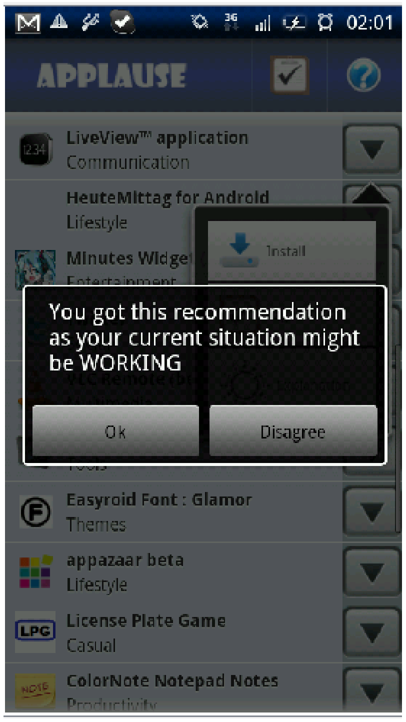
Application Package Structure



Application Screenshots

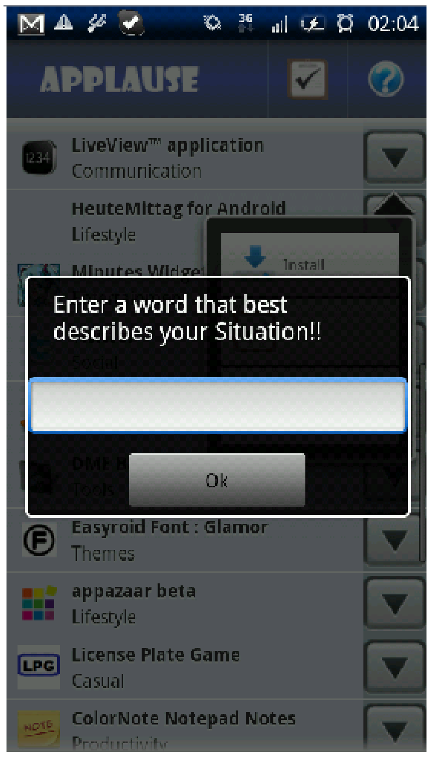
Recommendation List Recommendation Explanation



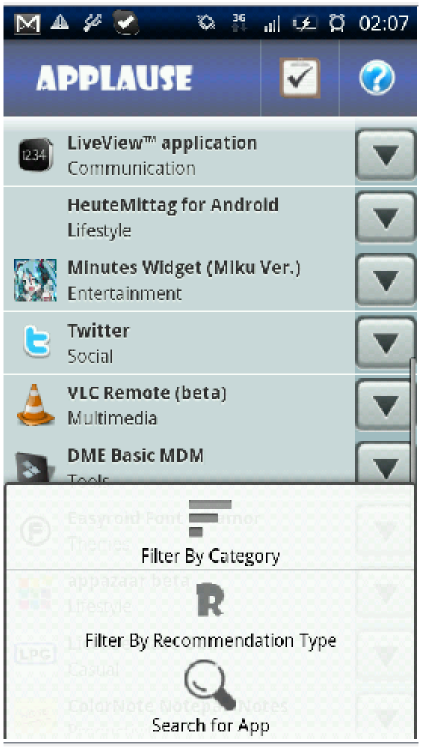


You got this recommendation as your current situation might be WORKING

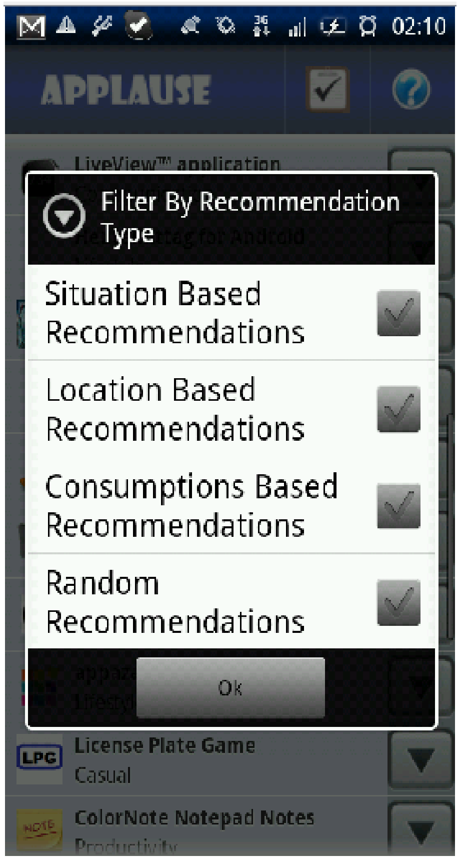
Situational Feedback Recommendation Filter



Enter a word that best describes your Situation!!



Filter By Recommendation Type Application Details



02:10

APPLAUSE

LiveView™ application

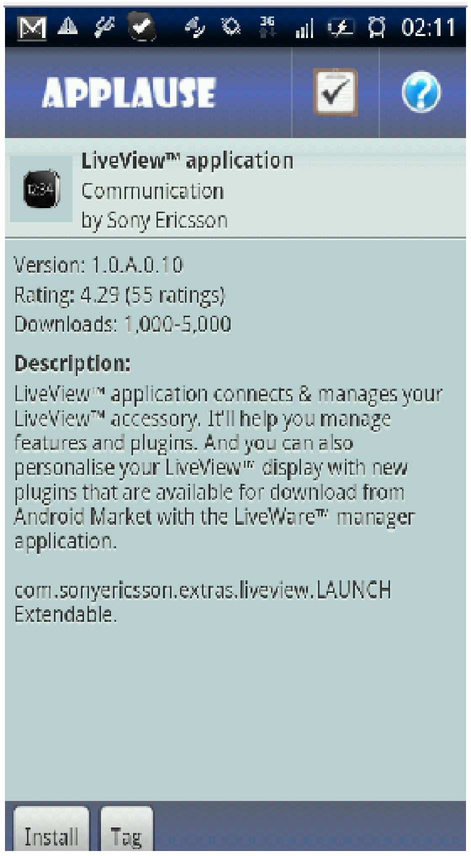
Filter By Recommendation Type

- Situation Based Recommendations ✓
- Location Based Recommendations ✓
- Consumption Based Recommendations ✓
- Random Recommendations ✓

Ok

License Plate Game
Casual

ColorNote Notepad Notes
Productivity



02:11

APPLAUSE

LiveView™ application
Communication
by Sony Ericsson

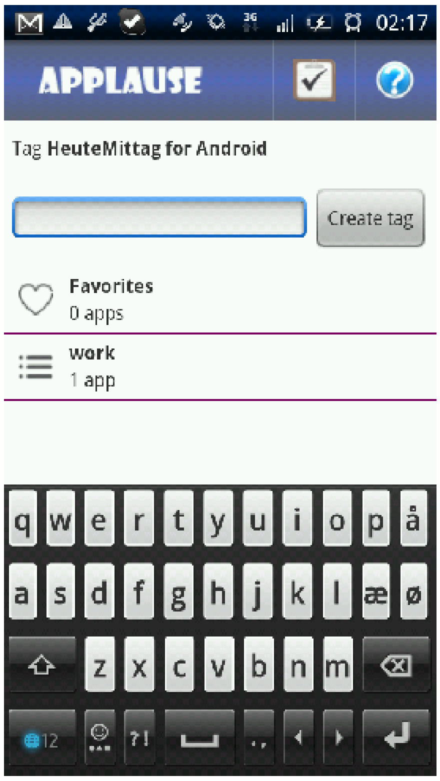
Version: 1.0.A.0.10
Rating: 4.29 (55 ratings)
Downloads: 1,000-5,000

Description:
LiveView™ application connects & manages your LiveView™ accessory. It'll help you manage features and plugins. And you can also personalise your LiveView™ display with new plugins that are available for download from Android Market with the LiveWare™ manager application.

com.sonyericsson.extras.liveview.LAUNCH
Extendable.

Install | Tag

Tag App Help Menu



02:17

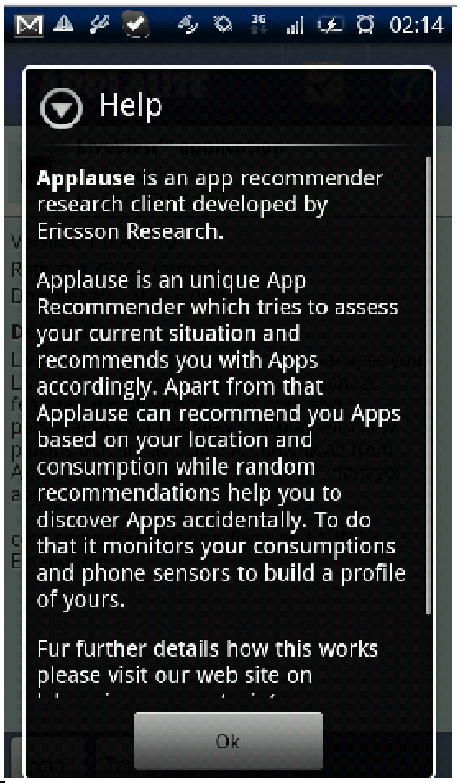
APPLAUSE

Tag **HeuteMittag for Android**

Create tag

Favorites
0 apps

work
1 app



02:14

Help

Applause is an app recommender research client developed by Ericsson Research.

Applause is an unique App Recommender which tries to assess your current situation and recommends you with Apps accordingly. Apart from that Applause can recommend you Apps based on your location and consumption while random recommendations help you to discover Apps accidentally. To do that it monitors your consumptions and phone sensors to build a profile of yours.

Fur further details how this works please visit our web site on

Ok

List of Android App Markets & App Statistics

Name	*Available apps*	*Device platform*	**Development platform(s)**
AndSpot	Unknown	Android	Android SDK
AppsLib	445 (October 20, 2009)	Android	Android SDK
BloomWorlds	Unknown	Android	Android SDK
Cellmania	Unknown	Multiple Android, BlackBerry OS, Flash Lite, iOS, Java,Symbian, Windows Mobile	Unknown
GetJar	68,625 (May 2010)[26] ~11k DLs/app	Multiple Android, BlackBerry OS, Flash Lite, iOS, Java,Palm OS, Symbian, Windows Mobile	Unknown
MobileRated	55,000 (December 2010)[28]	Multiple Android, BlackBerry OS, Java	Android SDK,Java ME
Handmark	Unknown	Multiple Android, BlackBerry OS, iOS, Java, Palm OS,Symbian, Windows Mobile	Unknown
Mobango	Unknown	Multiple Android, BlackBerry OS, iOS, Java, Palm OS,Symbian, webOS, Windows Mobile	Unknown
Handango	190,000[30]	Multiple Android, BlackBerry OS, Java, Palm OS, PSP,Symbian, Windows Mobile	N/A
explorePDA.com	1,500	Multiple Android, BlackBerry OS, iOS, Java, Palm OS,Symbian, webOS, Windows Mobile	Java ME, S60
MiKandi	Unknown	Android	Unknown
MobiHand	5,000 (June 3, 2009)[36]	BlackBerry, Palm, Symbian, Windows Mobile and Android	Unknown

Mobspot	Unknown	Multiple Android, BlackBerry OS, iOS, Java, Palm OS,Symbian, webOS, Windows Mobile	Unknown
Mobile2Day,Smartphone.net	140,000	Symbian OS, Palm OS, Windows Mobile, BlackBerry, Android, JavaME. webOS	N/A
PocketGear	140,000 (June 2010)[37]	Multiple Android, BlackBerry OS, Java, Palm OS,Symbian, Windows Mobile	N/A
AndroidGear	Unknown	Android	N/A
SlideME	1,860 (July 21, 2010)	Android	Android SDK

Reference

1. http://www.gartner.com/it/page.jsp?id=1480514

2. http://davidcrow.ca/article/7609/recommendation-engines'

3. D. Goldberg, D. Nichols, B. M. Oki and D. Terry, "Using Collaborative Filtering to Weave an Information Tapestry",Communications of the ACM 35 (1992), 61–70

4. Petteri Nurmi, Patrik Floréen, "Reasoning in Context-Aware Systems", 2004, available at : http://www.cs.helsinki.fi/u/ptnurmi/papers/positionpaper.pdf.

5. Dey, A. K. (2001). Understanding and Using Context. Personal and Ubiquitous Computing, 5(1), 4-7.'

6. Matthias Böhmer, Gernot Bauer: The Case for Context-Collaborative Filtering in Pervasive Services Environments. In: MobileHCI '09 - Workshop on Context-Aware Mobile Media and Mobile Social Networks. Bonn 2009, Germany.

7. Matthias Böhmer, Gernot Bauer, Antonio Krüger: Exploring the Design Space of Context-aware Recommender Systems that Suggest Mobile Applications. In: RecSys 2010 - Workshop on Context-Aware Recommender-Systems. Barcelona 2010, Spain

8. Przybilski, M., Nurmi, P., Floréen, P. "A framework for context reasoning systems".

9. Cheng , D. , Song , H., Cho, H. , Jeong , S., Kalasapur, S. , Messer, A., "Mobile Situation-Aware Task Recommendation Application", Proceedings of the 2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies, p.228-233, September 16-19, 2008.

10. Setten, M.V., Pokraev, S., and Koolwaaij, J. Context-Aware Recommendations in the Mobile Tourist Application COMPASS. Proceedings of AH. 2004, 235-244.

11. Adomavicius, G., and Tuzhilin, A. (2001c), "Extending recommender systems: A multidimensional approach" Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-01), Workshop on Intelligent Techniques for Web Personalization.

12. Karatzoglou, A., Amatriain, X., Baltrunas, L., and Oliver, N. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In Proceedings of RecSys. 2010, 79-86.

13. Adomavicius, G., Sankaranarayanan, R., Sen, S. , Tuzhilin, A. Incorporating contextual information in recommender systems using a multidimensional approach, ACM Transactions on Information Systems (TOIS), v.23 n.1, p.103-145, January 2005

14. Jonathan, L.H., Joseph, A.K., and John, R., "Explaining collaborative filtering recommendations". In proceedings of the 2000 ACM conference on Computer supported cooperative work (CSCW '00), ACM

15. Davidson C., Moritz S..: “Utilizing Implicit Feedback and Context to Recommend Mobile Applications from First Use”, CaRR '11 Proceedings of the 2011 Workshop on Context-awareness in Retrieval and Recommendation.

16. Berkovsky, S., Aroyo, L., Heckmann, D., Houben, G.J., Kroner, A., Kuflik, T., Ricci, F.: "Providing Context-Aware Personalization through Cross-Context Reasoning of User Modeling Data", 11th International Conference on User Modeling, pp. 2-7, Publ. User Modeling Inc. (2007)

17. Ian Watson, Farhi Marir, “Case-Based Reasoning: A Review”, http://www.ai-cbr.org/classroom/cbr-review.html

18. Y. Asakura a, E. Hato, “Tracking survey for individual travel behaviour using mobile communication instruments”, Transportation Research Part C 12 (2004) 273–291

19. Falaki, H., Govindan, R., Estrin, D., “Challenges of Smarter Power Management on Smartphones”; http://www.cs.ucla.edu/~falaki/pub/EnergyChallanges.pdfhttp://www.cs.ucla.edu/~falaki/pub/EnergyChallanges.pdf