NTNU
Norwegian University of
Science and Technology

Recommendation of Attractions and Activities

Using Collaborative Filtering and Implicit
Feedback

Magnus Settemsli Mogstad
John-Olav Storvold

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Herindrasana Ramampiaro, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Title:	Recommendation of tourist attractions
Students:	Magnus Settemsli Mogstad, John-Olav Storvold

Problem description:

In a modern world, people travel more. They are busier scheduling their day to day activities, and they value others' opinion now more than ever. Intelligent recommendation systems are becoming an indispensable tool for facilitating decision making for the users, whether it is online shopping, recommending new music or highlight movies you have yet to watch.

The current mindset for recommendation systems is to tailor their ability to provide excellent recommendations within their targeted field. This project aspires to design and develop a proof-of-concept that offers personalized recommendations within multiple domain areas, which not only is achievable but frankly what people currently is missing in their lives. The proof-of-concept will give personalized recommendation within movies, museums, music events, nightlife, and restaurants.

Responsible professor:	Heri Ramampiaro, IDI
Supervisor:	Heri Ramampiaro, IDI

Abstract

Recommendation systems are becoming more and more popular and are introduced to new domains all of the time. Therefore, the purpose of this master's thesis is to investigate if a recommender system into the domains of attractions and activities is plausible both to create and if it is viable towards the end user. We aim to proof-of-concept a recommendation system which utilizes collaborative filtering, implicit feedback, and user profiling. Further, an experiment is conducted with external users. The experiment will collect data from the users, their thoughts of the service and if they found the service viable and the recommendation given to be credible.

In the first part of the thesis, there is an empirical study of the methods and techniques used in recommendation systems. This study includes methods used in modern recommendation services nowadays and is used as a guideline when creating and discussing the service. The second part of the thesis undertake the requirement, design and implementation parts where we decide which methods to utilize, how we created the service and what technologies used. Lastly, we discuss and evaluate our findings regarding the service's viability and credibility using the results obtained in the conducted experiment.

The results of the proof-of-concept experiments show that the participants found the system credible, and the thesis will argue that such a recommendation system is viable.

Sammendrag

Anbefalingsystemer blir mer og mer populært og det blir introdusert innenfor nye domener hele tiden. Formålet med masteroppgaven er å undersøke om det er mulig å lage et anbefalingssystem som fungerer på flere domener innenfor attraksjoner og aktiviteter, og for å se om et slikt system er noe sluttbrukerne synes fungerer. Målet med oppgaven er å bevise at et prototype av slikt anbefalingsystem fungerer ved hjelp av samarbeidsfilter, implisitt tilbakemeldinger og brukerprofiler. Et eksperiment er også gjennomført med eksterne brukere. Igjennom eksperimentet samler vi inn data fra brukeren, deres tanker om systemet, om de synes systemet fungerte og at anbefalingene deres var troverdig.

I den første delen av masteroppgaven er det gjennomført en empirisk studie om metodikken og teknikkene som er brukt i anbefalingssystemer. Denne studien inkluderer metoder og teknikker som er brukt i dagens anbefalingsystemer. Studien er veiledende for design og implementasjonen av systemet. Den andre delen av oppgaven omhandler krav til systemet, design og implementasjonen gjennomført, og hvilke metoder og teknologier som ble brukt for å lage systemet. Til slutt så evaluerer og diskuterer vi våre funn om tjenestens troverdighet og funksjonalitet ved hjelp av resultatene som ble innhentet i et eksperiment.

Resultater fra eksperimentet viser at deltagerne fant anbefalingene til prototypen troverdige, og oppgaven argumenteres det for at systemet også fungerer.

Preface

This thesis concludes our Master of Science education in Computer Science at The Norwegian University of Science and Technology (NTNU) in Trondheim. The thesis was performed throughout our 10^{th} semester, spring 2016, at the Department of Computer and Information Science

Acknowledgments

First and foremost, we would like to thank Heri Ramampiaro for being our supervisor. We are grateful that we could come to you with any questions during this time. We would also like to than our friends and families who have helped us both by participating in our experiment and help us distribute our questionnaire and lastly helping us with proofreading our thesis.

Contents

List of Figures

List of Tables

List of Algorithms

List of Listings

List of Acronyms

API Application Programming Interface.

BIC Bayesian Information Criterion.

BIRCH Balanced iterative reducing and clustering using hierarchies.

CF Collaborative Filtering.

DOM Document Object Model.

HTML Hyper Text Markup Language.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

k-means K-nearest neighbor.

NLP natural language processing.

npm Node Package Manager.

NTNU Norwegian University of Science and Technology.

RQ research question.

SNE Stochastic Neighbor Embedding.

TCP Transmission Control Protocol.

t-SNE t-Distributed Stochastic Neighbor Embedding.

UDP User Datagram Protocol.

UI User Interface.

WWW World Wide Web.

Chapter 1
Introduction

Most of us have been in a situation when traveling where we have some time to kill or a new place to discover. In the digital age we live in now, it is common to use the internet to explore the possible sites to visit, who are in your vicinity. We have applications like TripAdvisor, Stay.com, and others that try to give you a list of tourist attraction in the city you are currently located, but none of them offer the possibility of personalized recommendations of attractions or activities at your location where the recommendations are based on your preferences and history from other vacations.

1.1 Motivation

Every day people struggle from time to time when opting what activities they would like to engage in. Sometimes these activities present themselves fairly naturally; either from passively getting to know of if by word of mouth to something more actively such as going online and searching for an event you would like to attend. There is nothing wrong with using either of these ways to getting to know your options, but there is one thing that neither of these methods does, and that is guaranteeing that you will find what you are looking for nor ensure that the time spent researching will pay off regarding attending said activity. We want to check if it is possible to give personalized recommendations for attractions and activities. There are a few applications out there that try to do the same. What differentiates our goal from what they offer is that would like to combine the knowledge of what we know about the person and the activities around the person's location to give the user recommendations, and display the locations of the results on a map for the user. We would like to find out if it is possible to create a system which will make intelligent decisions to give people recommend activities across multiple domains so that the person will spend more of his or her time doing said activities than looking for them.

Further, we hope that by checking if such a system is viable and that the required time spent on scheduling an evening will reduce drastically and at the same time be part of how people would go about when planning a weekend or a trip. Hopefully, the research we will conduct whether such a system is viable will turn out positive, and we will get to a point where helpful suggestions will be given to the people using the system during an experiment.

Moreover, we want to develop a fair strategy to recommend multiple domains given how diverse and different some of the domains are from each other. Previous research has mainly focused on one specific domain, i.e. e-commerce, fashion, books, and movies. A good recommendation algorithm for our specific needs has to be general and modular such that there is no need to maintain several independent systems. We are motivated by the idea to have one particular system in place that easily can be modified to add support for other domains in the future. Further, previous works mention many different ways to achieve this both using explicit and implicit feedback. For some domains, explicit feedback will prove to be the better way to give good recommendations. In other cases, implicit feedback may prove to be the better both for the user and the resulting recommendations.

1.2 Challenges and Current Situation

It is quite a challenge to proof-of-concept a service that provides the users with recommendations of which attractions and activities to visit. The difficulty of our task comes from the plurality, and not from the fact that there are no solutions to it. There exists a lot of ways to create a recommendation system within any single domain such as movies, books or music. Secondly, restricting what domains we were to work on was a challenge itself. The importance was to limit the domains to work with while at the same time maintain that degree of generality we are aiming for. Lastly, the time limitation combined with the broad scope of the task is also a challenging factor. With those in mind, we have limited our recommendations within the domains movies, museums, music, nightlife and restaurants.

Further, it is a challenge to find a dataset containing attractions and activities; the dataset must also hold metadata about the entities, which can be used to classify them into categories and also distinguish them from similar ones by looking at their metadata. An example of this would be two museums where one is about ancient Egyptian art whereas the other one is a modern contemporary art. They are both museums and both art, but they are vastly different from each other. In such a case, it is important to be able to have a dataset that has a rich source of information. There are a few companies around with such databases, although among the same ones they have restrictions on usage. In light of this, we took advantage of some of the more sturdy Application Programming Interface (API) available to us, such as

Google Places,[1] Yelp,[2] and Foursquare.[3]

1.3 Personalized Recommendations

We would like to proof of concept a service which offers users a personalized recommendation of attractions and activities. We will do this by using implicit feedback and by creating a modular recommendation system based on Collaborative Filtering (CF), and utilize the user profile with the users preferences to help the system give recommendations in the early life of the service.

The value of our research will be to attract attention to a currently unexplored research field within recommender systems. Surprisingly, we do not know of any existing services or applications that provide and recommend travel information. We believe this might bridge a gap between tendencies of how people travel, and a lack of service for it.

1.4 Research Questions

From the problem specification, we have the following main research question (RQ).

– RQ: How to build a proof-of-concept service to provide recommendations of attractions and activities?

We have broken down the main research question into the following RQs to help us with our research.

– RQ1: How are the recommendation techniques in today's recommendation systems?

– RQ2: How to develop techniques used to recommend attractions and activities?

– RQ3: How to provide up-to-date information necessary for such a recommendation system?

– RQ4: How to implement a proof-of-concept recommendation service?

[1]https://developers.google.com/places/
[2]http://www.yelp.com/
[3]https://foursquare.com/

1.5 Outline of Report

In this chapter, we presented our motivation for our work within this area of recommendations and the proof-of-concept research we have done. We also mention what challenges we have faced and try to specify our problem specification into several research questions. In chapter 2, we present a survey of previous research within the domain of recommendations and the different techniques used when creating a recommender system. Chapter 3 presents our approach, requirements and design for the application and discusses what methods to use in the different modules of the application. In chapter 4, we talk about the specifics of our implementation and mention key technologies and frameworks used to fulfill the system requirements. Furthermore, we discuss how we have implemented the methods employed in the recommendation service and modules.

Chapter 5 includes our evaluation and discussion of our recommender system. We walk through the results from our user experiment. We also discuss some of the choices we have made during the length of the research. Finally, chapter 6 concludes our work and summarizes what we have learned. We talk about the existing limitations of the implementation and compromises taken. Further, we list future promising work to complement the work already done and our final thoughts on the research.

Chapter 2 Survey

In this survey chapter, we will try to review relevant literature used in our thesis. We will talk about the options we found when researching the possibilities and what their strengths and weaknesses are. We will talk about how and where to get datasets from, methods used to create recommendation systems, other recommendations systems, clustering methods and dimensionality reduction of data.

2.1 Datasets

For the tourist attraction recommendation system we will need to get hold of data about the different tourist attractions, but the system will also have to be able to keep up with the updated information about attractions to continue to give viable attraction recommendations since the existing information may become outdated. May it be due to new businesses replacing existing businesses, or that the original company no longer exists. There's a vast range of available options as to where you could get your data from. We would like to mention the available options briefly and why some of them are less relevant than others.

2.1.1 Web Crawlers

A web crawler was initially intended to search the World Wide Web (WWW) systematically for the purpose of web indexing. However, you can tailor a web crawler to target specific websites to extract information, which you then can use to build a database. Using a web crawler is an excellent way to gather information which is updated and a good starting point for our service. With the utilization of this approach, we can face the problem that the websites we crawl data from, does not contain updated information, and therefore our database might not contain up-to-date information. Luckily, if we manage to create a useful web crawler to begin with, we should be capable of customizing that web crawler to search for updated information rather than all information.

2.1.2 Publicly Open Datasets

The second available option is to make use of a publicly available dataset. There are a lot of them, and many are relatively recent ones and of a significant size. However, a big concern is that even if they are recent, they still have a few years under the belt, meaning they no longer can be regarded as updated information. Their initial purpose is in a lot of the cases to be used for scientific research, and for that purpose they are a great starting point. Notably in the early phases of a new project where the task is a proof of concept to see if the proposed system is viable, where viability is the ideal outcome rather than a complete service ready to be deployed. If the latter is the case, you will have to find a way to make use of the existing database and take it from there. It may be active users that are keeping your content updated by giving feedback on whether the data is correct. Google currently gathers information about businesses by questioning individuals who have been there through Google Maps.

2.1.3 Third-Party Services

An approach some recommendation services that are in production seems to favor the use of existing third-party services and databases. There are a lot of companies that have built their database for themselves over the years and offer their information to others with the use of web API. A few websites to mention is TripAdvisor, Yelp, Google Places, Foursquare, Songkick and so forth. The information you work with is then managed by a third-party, relieving you from some of the responsibilities they face when providing their service. However, the downside is that you will have to work on their terms and you are not in control of the data should you want to be. It might be costly to use their service based on the traffic on your service. They may also be unavailable for a given moment for which your service will render useless. Speaking of useless; What if they want a divorce? You will have to move on and find a new service that fits your needs. That could potentially be quite costly, and depending on which line of business you were working with, another third-party service might not even exist. Lastly, you will have to deal with a delay as you will have to ask their service for information every time a user wants a recommendation.

2.1.4 Existing Dataset

What is the best source of data for a recommendation system, and which solution gives the most persistent and updated data? The web crawler is not an optimal solution because you never know if the attractions in your dataset are currently up-to-date and, in business, or closed, without crawling the data from a data provider with persistent data. Public datasets are not a good solution if your system is to be used as an online recommendation system. A user that wants real-time information, and you can not guarantee that the data is useful at this point. Is the restaurant still in business, or have they closed down since last time the crawler checked?

Furthermore, there does not exist any data set to our knowledge which provides us with the data we will need to give recommendations in all of the domains used in our system.

Therefore, we have come to the conclusion that the best solution for the recommendation system is to use a third-party service data set to get information which is persistent and updated. The different services the system will use for recommendations are Foursquare, Songkick and scrape data from Google Showtime.

2.2 Previous Works

Recommendation systems are a special type of information filtering systems. These systems filter items from a large collection of data, and produce output data that the user is likely to find interesting or useful[39]. There exists a lot of research in the area of recommendation systems, but most of the research is within the field of e-commerce. Almost every big market chain uses recommendation systems nowadays. The reason for this is to give the users knowledge about products that they might not have found on their own, and that is closely related to other products bought or looked at.

The two main methods used for feedback in recommendation systems are implicit feedback and explicit feedback. Implicit feedback is inferred from the users' behavior such as what attractions they look at, the duration they spent looking at the attraction, and if they want to go or already have been. Explicit feedback is when the users are asked to rate a place they have been. There exist many diverse ways to measure using explicit feedback. One of the methods is binary rating. Google Places allows their users to leave a rating between 0-5 for their service.

2.2.1 Content-based Filtering

Content-based filtering is a recommendation method that recommends items to the user based on the content of the item. An item is recommended to the user if the content of the item is related to the users' likes and interests. Content-based filtering uses training data which consists of items users found interesting. The items in the training data all have distinct attributes. The attributes of the item specify the item's class based on the item rating by the users, or on data collected via implicit or explicit feedback. The items used in the training data are usually represented as a vector of n different values $\mathrm{X}_n = (x_1, x_2, ..., x_n)$. The different attributes in the item vector can have different types of values like binary, nominal or numerical values, which is derived from the content of the item or the information about the users preferences. With the training data, the learning methods task is to classify any new items with a positive or a negative by returning a binary value or a numerical value.

A system built for content-based filtering finds items to recommend to the user based on a correlation between the users' preferences and the item. The system recommends items by comparing the profile of the user with the content of each item in the collection. Items the user find interesting can be determined by using both explicit or implicit feedback. If the system uses explicit feedback, it requires the user to rate the item on a set scale. With implicit feedback, the user interest is inferred by the users actions. For instance, the length of time the user looks at the item. The implicit feedback method improves the user experience, but it is harder to implement and utilize.

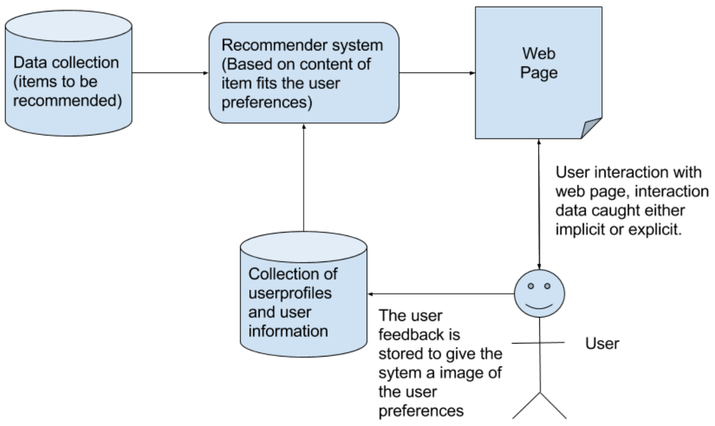

Data collection (items to be recommended)
Recommender system (Based on content of item fits the user preferences)
Web Page
User interaction with web page, interaction data caught either implicit or explicit.
Collection of userprofiles and user information
The user feedback is stored to give the sytem a image of the user preferences
User

Figure 2.1: Content-based recommender system architecture

2.2.2 Collaborative Filtering

CF is a popular recommendation algorithm that utilizes user ratings and user behavior to predict what the user likes. CF works best when the user is required to rate the different items, and through item ratings CF can predict what items the users will appreciate[52]. The method focuses on matching people based on their preferences and finding similar users to give recommendations.

A successful collaborative filtering system needs to meet the following conditions:

1. Not only a large user base, but also users who have a diverse taste
2. It must be easy to represent what you like to the system.
3. The algorithms must be able to identify users with similar interests.

In order for CF to find an approximation of what the users' like and dislike, CF requires the users to rate varied items. The system uses these ratings to match an individual user with other users in the system. Additionally, the ratings give a clue of what the users taste is in the domains the system contains. Even the unexplored domains for a particular user is handled judging rated items of adjacent users. The final phase is to recommend the things the users' nearest neighbors have rated, but have not rated themselves.

A challenge with CF is how to give weights to the items rated by your neighbors. The user should be able to choose if they do not like recommended items, this gives the system a better understanding of the users preferences and can give lower weights to neighbors who like items the user do not like. Amazon.com uses item-item collaborative filtering extensively to give the user recommendations on new products they might find interesting. Amazon uses the Item-Item recommendation algorithm[35] because it is scalable over a large customer base and product catalogs. Moreover, it makes good recommendations to the users regardless of the number of items rated and bought.

Algorithm 2.1 Amazon.com Item-Item recommendation

```
procedure ITEM-ITEM RECOMMENDATION
    for each item in product catalog, $I_1$ do
        for each costumer C who purchased $I_1$ do
            for each Item $I_2$ purchased by costumer C do
                Record that a customer purchased $I_1$ and $I_2$
            end for
        end for
        for each item $I_2$ do
            Compute the similarity between $I_1$ and $I_2$
        end for
    end for
end procedure
```

Cold Start Problem

A problem with collaborative systems is that it suffers from a problem known as the "cold start" problem. "Cold start" describe a potential problem in recommendation systems in the early stages of system's lifespan. In this initial phase, the system has insufficient information about neither the users nor the items, which can result in poor recommendations[47]. It might better not to give a recommendation to the user at all if the data about the users preferences is too sparse to give a good recommendation.

There are three types of cold start problems:

- Item: A new item has been added and does not have enough ratings to be recommended to anyone.
- User: A new user has started using the system, but their likes and dislikes are not known yet.
- Community: The problem with creating a recommendation system due to the challenges in collecting enough data to give good recommendations.

User-User Collaborative Filtering

The User-User Collaborative Filtering is one of the first CF methods which recommends items to the users automatically. User-User recommendation algorithm finds other users who have the same taste and behavior as the current user. Further, those items they have rated are used to predict which items the current user would like. The User-User CF requires user models made up of a two-dimensional matrix; users on one dimension and items on the other. The model holds all the ratings the users have given. Finally, CF uses a similarity function s: UxU $\rightarrow \mathbb{R}$ to calculate the similarity between users.

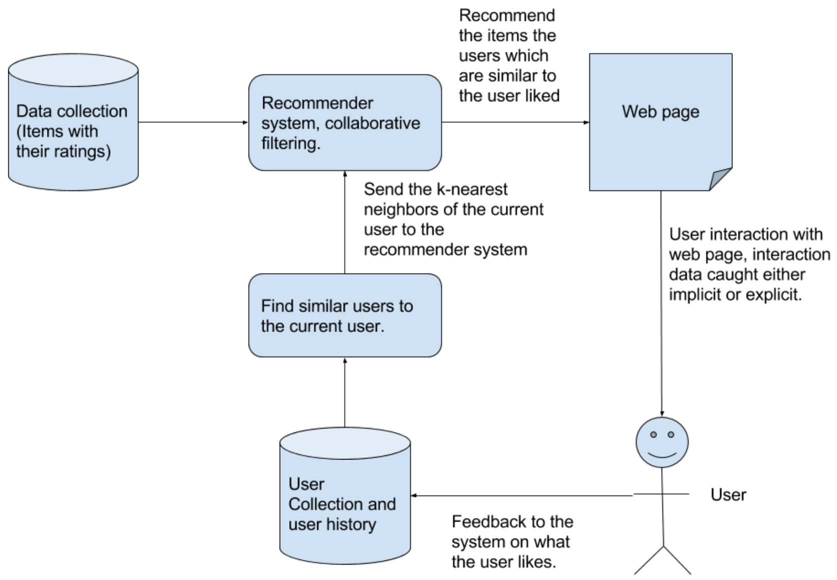

Data collection (Items with their ratings)
Recommender system, collaborative filtering.
Recommend the items the users which are similar to the user liked
Web page
Send the k-nearest neighbors of the current user to the recommender system
User interaction with web page, interaction data caught either implicit or explicit.
Find similar users to the current user.
User Collection and user history
Feedback to the system on what the user likes.
User

Figure 2.2: A user-user collaborative filtering system architecture

Pearson Correlation

The Pearson Correlation is a method to compute the statistical correlation between two users[28]. It uses the ratings to determine how closely related in taste the two users are. Equationp. 2.1 represents the formula for Pearson Correlation.

$$s(u,v) = \frac{\sum\limits_{i \in I_u \bigcap I_v} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum\limits_{i \in I_u \bigcap I_v} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum\limits_{i \in I_u \bigcap I_v} (r_{v,i} - \overline{r}_v)^2}} \tag{2.1}$$

Pearson correlation is not the correct method to utilize if the users have few rated items in common. It is typical to set a threshold to an amount of shared items both users have ranked before calculating the Pearson correlation between them. The correlation formula will return a value between -1 and 1, where -1 is the total negative correlation, 0 is no correlation, and 1 is the total positive correlation. In other words, if two users have identical taste, the correlation between them equals 1. Previous experiments have shown that a threshold of 50 is useful to improve prediction accuracy. You can see an example using the Pearson correlation in appendix A.1.

`Spearman Rank Correlation`
The Spearman rank correlation is another method to find similar users from their ratings and actions. The Spearman correlation ranks the items a user have rated in a list ranking. For instance, the highest rated item has a rank of number one. Naturally, the second highest rated item, no matter how close or far ranked comparatively to the highest rated item, will have a rank of number two. Furthermore, items that share the same rank will have an average rank accordingly to their positions. The calculation of Spearman rank correlation is then equal to Pearson correlation with the exception of using the items list ranking instead of their ratings.

`Cosine Similarity`
The Cosine similarity function is a vector-space approach that uses linear algebra instead of a statistical approach to find similar users. The users are placed in a |I|-dimensional vector space, and the similarity between the users is calculated by determining the distance between the two user vectors in the vector space.

$$s(u,v) = \frac{r_u * r_v}{||r_u||_2||r_v||_2} = \frac{\sum_i r_{u,i} * r_{v,i}}{\sqrt{\sum_i r_{v,i}^2}\sqrt{\sum_i r_{v,i}^2}} \tag{2.2}$$

The items users have not rated is set to the value of zero as they then it will drop out of the numerator. The cosine similarity is equivalent to the Pearson correlation when the two users compared have rated the same items.

`Other Similarity Functions`
There has been an effort in both finding and using different similarity functions, but they are not commonly used. The Pearson correlation produces the best results when obtaining users who have similar taste.

Item-Item Collaborative Filtering

The User-User CF approach is an efficient method to use, but it suffers performance scalability wise when the user base gets larger because searching for neighbors takes $\mathcal{O}(|n|)$ time to compute[28]. A way to extend the usage of collaborative filtering to a system with a large user base, and for it to realistically utilized on an e-commerce site, the Item-Item collaborative filtering method was developed to face the scalability problems. This method is one of the most used collaborative filtering techniques used today in recommendation systems. Item-Item collaborative filtering uses similarities between the rating pattern of items to give users recommendations. If two items get the same ratings from different users, then it is likely that users who have the same taste will like the item. This method is similar to a few content-based approaches, but the similarity between items are deducted from the ratings the users give it and not from the item data.

The Item-Item collaborative filtering method does not solve anything by itself as it still needs to find similar items to give the users a recommendation. The method finds it reasonable to pre-compute a matrix with data that tells how similar the different items are. When users change ratings, the data in the matrix shifts so it will need to be computed again within a reasonable time. Even if the computation is not the newest, it is likely that users will receive good recommendations from the system. Cosine similarity, conditional probability, and Pearson correlation are the two most used methods to calculate the similarity between two items.

`Cosine Similarity`
Similarly to User-User collaborative filtering, Item-Item collaborative filtering can also utilize the cosine similarity function (equation. 2.2) to calculate similarity. This method is the one most used when it comes to finding similar items. It is simple, fast and produces good results.

As seen in [40] we know that the cosine similarity seems to work better than other similarity functions like the Pearson correlation(section 2.2.2) in high-dimensional data. However, the cosine similarity is also somewhat affected by the curse of dimensionality.

`Conditional Probability`
Conditional probability is used when trying to find similar items from unary ratings such as shopping history. This method uses normalization that hinders the ability for it to produce true predictions, but the method remains relevant by using pseudo-predictions.

$$s(i,j) = \frac{Freq(i \wedge j)}{Freq(i)(Freq(j))^{\alpha}} \tag{2.3}$$

`Pearson Correlation`
The Pearson correlation can also be used when finding the similarity between items in item-item CF. But it does not produce as good results as cosine similarity when finding similar items, as it does when used in user-user CF.

2.2.3 Hybrid Approach

In an attempt to find better recommendation systems, researchers have tried to combine two or more recommendation algorithms into a hybrid solution. In some of the experiments, a hybrid recommendation system has outperformed an individual recommendation algorithm. A hybrid system can be the solution in complex recommendation systems to cover individual algorithms weaknesses. For instance, Item-Item CF struggles to recommend an item when no one has rated it. In this scenario, a content-based algorithm, which works with items without a rating, can be used in combination with Item-Item CF. Most common hybrid recommendation systems are built up using collaborative filtering and some other recommendation method[24]. Burke[24] gives an analysis of hybrid approaches, and groups them into seven different categories.

1. Weighted
The weighted method estimates a score of a recommended item by using all of the available recommendation methods available in the system. It combines the generated recommendations to give the user a list of recommended items.

2. Switching
The switching method switches between different recommendation methods and ultimately ending up with using the method that is expected to give the best results in the current situation.

3. Mixed
The mixed method presents the recommendation results from all the methods used. The results are not shown in on a single list as with weighted, but each method displays its results.

4. Feature combination
The feature combination method uses different features from many recommendation algorithms to create one recommendation algorithm.

5. Cascade
The cascade method runs one recommendation algorithm. The second algorithm uses the output of the first algorithm as its input. The produced output of the second algorithm is the recommended items to the user.

6. Feature augmentation
Feature augmentation is when a recommendation method uses the output of another algorithm as its feature input.

7. Meta-level
A recommendation algorithm train a model, the model is used as input for another recommendation algorithm. The output from the second recommendation algorithm is the recommended items for the user.

A hybrid method can help with some of the known problems the classical recommendation techniques. Both content-based filtering and CF shows a decrease in performance when used in a hybrid approach. The performance decrease is due both methods need a database with ratings and users. Nevertheless, even with a loss in performance, the hybrid approach is still very popular as a result of the robustness of the hybrid approach when it comes to the recommendation.

Technique	Advantages	Weaknesses
Collaborative filtering(CF)	Can identify cross-genre niches. Domain knowledge not needed. Adaptive: quality improves over time. Implicit feedback sufficient.	New user ramp-up problem. New item ramp-up problem. “Gray sheep” problem. Quality dependent on large historical data set. Stability vs plasticity problem.
Content-Based	Can identify cross-genre niches. Domain knowledge not needed. Implicit feedback sufficient.	New user ramp-up problem. Quality dependent on large historical data set. Stability vs plasticity problem.
Demographic	Can identify cross-genre niches. Domain knowledge not needed Adaptive: quality improves over time	New user ramp-up problem. Quality dependent on large historical data set. Stability vs plasticity problem. "Gray sheep" problem. Must gather demographic. information.
Utility-based	No ramp-up required. Sensitive to changes of preference. Can include non-product features.	User must input utility function. Suggestion ability static (does not learn).
Knowledge-based	No ramp-up required. Sensitive to changes of preference. Can include non-product features. Can map user needs to products.	Suggestion ability static (does not learn). Knowledge engineering required.

Table 2.1: Tradeoffs between recommendation techniques [24]

2.2.4 Other Recommender Systems

As mentioned earlier, it does not exist much if any research in the area of recommending tourist attractions using CF or content-based filtering. Luckily, there exists a lot of research in the field of recommendation.

Recommender systems are defined by Pei Wang as “A system that has as its main task, choosing certain objects that meet the requirements of users, where each of these objects is stored in a computer system and characterize by a set of attributes”.[54]

Personalized Fashion Recommender System

In “Learning to Rank for Personalized Fashion Recommender Systems via Implicit Feedback”[41], the research focuses on giving the user a recommendation of which clothes the user should buy because it fits their style. The challenges this system faced was the problem of clothes popularity, time and cultural grouping. The method developed used different ways to gather data about what the user likes. Further, the methods they used was to find the user’s interests by recording the products the user looked at, keeping a database record with the clothes the user have marked as loved, or the clothes the user marked as a purchase. To register this data, they used implicit feedback from the users interaction with an application developed to gather data about users cloth interest. The data collected is used to analyze the recommendations given by the system.

Even though implicit feedback is popular when creating a recommendation system, it has some challenges. Among other there is no way to give negative feedback, and the different types of interaction need to be combined into a single numerical value. The fashion recommendation system focuses on three distinct user interaction events; click, wants and purchases. Events are naturally ordered; wants counts less than the purchases. The system takes the seasons, trends, price, and popularity into account when giving the user a recommendation. It also gives items a penalization function by giving each event a particular value inside the range of possible scores. The penalization is a way to give items a negative feedback when using implicit feedback.

The fashion system tries three different techniques to recommend clothes to the users.

Most Popular: Selects the most popular items, then uses dithering to randomize the recommendation given to the users.

Binary Recommenders: Uses binary data for user interaction, and applies two algorithms k-nearest neighbor item-based collaborative filtering and Bayesian Personalized Ranking (BPR).

Non-binary Recommenders: Bases the recommendation of clothes on the inferred implicit scores. It uses the alternating least-squares with weighted λ-regularization (ALS-WR) originally developed for the Netflix competition by using cross-validation and k-nearest neighbor user-based collaborative filtering with cosine similarity.

Recommender Systems Applied to Electronic Books

Users tend to seek for recommendations from others who have previously had the same needs[48]. Therefore, to gather data about user interests they started by defining a collection of implicit parameters, comparatively analyzing their values, and measuring their correlations. They infer the grade of interest that users may have for certain items while interacting with an electronic book.[42] The process allowed them to convert implicit feedback into explicit ratings that help with making more precise recommendations. The use of implicit feedback from the user has its grounds in that explicit feedback alter the user's regular navigation and reading patterns, due to the need to stop and rate the different books. To be able to give correct recommendations at any time, a recommendation system requires continuous learning about the user's profiles.

The research focuses on books in different categories; each category contains ten photo books, and each book has ten pictures. Thus, by adding a transparent layer to the system they could record the user's interactions with the system, to capture the implicit parameters and determine the number of times a user looks at the different categories. Each time the user looks at specific content or items, the system utilizes implicit feedback to gather the necessary data. You can measure more precisely the interactions the users have with the system using such layer. Allowing the system to know which categories, content or items the users visited, and the time the user spent looking at different content.

These are the measured parameters the system recorded.

1. Duration of the session: Indicates how long the user stays connected to the system, and how long the user interacted with the content.

2. Number of clicks: Measures how many clicks the user used to evaluate the content.

3. Reading time of a content: Describes how long a user takes to read or view the content. It is important as it determines the user's interest based on the average time spent reading or viewing some content.

4. Reading time of a category: Determines how long a user spends reading content in the different categories.

5. Number of accesses to a category or classification: How often a user have visited one category.

6. Number of comments: Is used to determine the general interest in the content, based on the amount of comments by the user.

7. Number of recommendations to a friend: A number of recommendations indicate the interest of users of the content basing on some recommendations. If a user recommends some content to another user, he believes the content will be attractive to the user.

The study ended up with different relations used to give the user a recommendation. It observed that all the explicit data they gathered about content was extensive. The reason being users who liked a book rated it, while users who did not like it did not bother rating it.

Building and evaluating a location-based service recommendation system with a preference adjustment mechanism

The mobile commerce is a developing trend due to the success of e-commerce and the development of wireless technologies over the recent years. The system goals are to establish a location-based information recommendation model. Thus, by integrating the attributes of user geographical location and personal preference[33]. The location-based service should provide to the user based recommendation based on their specific geolocation. A source of the data used in the location-based service is the user profile content; the user profile content contains values that describe the user interests and preferences. In content-based recommendation, the user profile is matched to the contents of different items, and the degree of similarity is calculated to determine if the item should be recommended to the user. In the location-based service recommendation, the model is divided into registration module, recommendation module and preference adjustment module, and each of the modules is interacting with each other through databases.

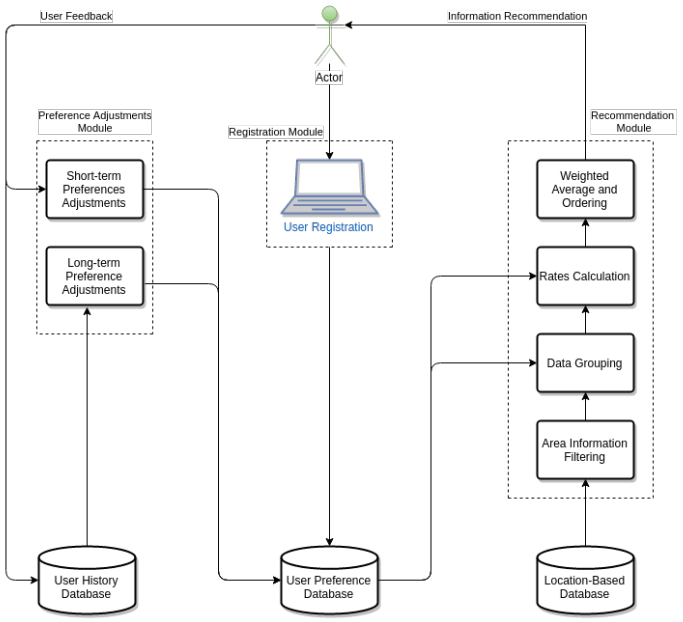

User Feedback
Information Recommendation
Actor
Preference Adjustments Module
Registration Module
Recommendation Module
Short-term Preferences Adjustments
Long-term Preference Adjustments
User Registration
Weighted Average and Ordering
Rates Calculation
Data Grouping
Area Information Filtering
User History Database
User Preference Database
Location-Based Database

Figure 2.3: The architecture of a location-based service recommendation model (LBSRM) [33]

`The Registration Module:`
Comes into play when the user creates a user profile. The user profile is used as a basis for the initial recommendation since the user has no other preferences, and the information in the user profile is used since there is no history stored in the database.

`The Recommendation Module:`
The recommendation module filters the information on the location of the user, and then it conducts the recommendation process by ranking the different items in the area. Learning of the users preferences is done when the feedback from the usage of the system is received. This is archived with two adjustments. Firstly, a short-term adjustment is performed with feedback gathered from the user. Secondly, a long-term adjustment is performed over time by using Bayes' decisions to predict the users' long-term preference.

The recommendation results are displayed to the user with a top-N method, where the best result is first, and the user can browse the item attributes item by item. The users satisfaction with the recommendation system is measured in the precision of the recommendations. The precision equation used in the system is as follows:

$$Precision = \frac{Correct \cap Recommended}{Recommended} \tag{2.4}$$

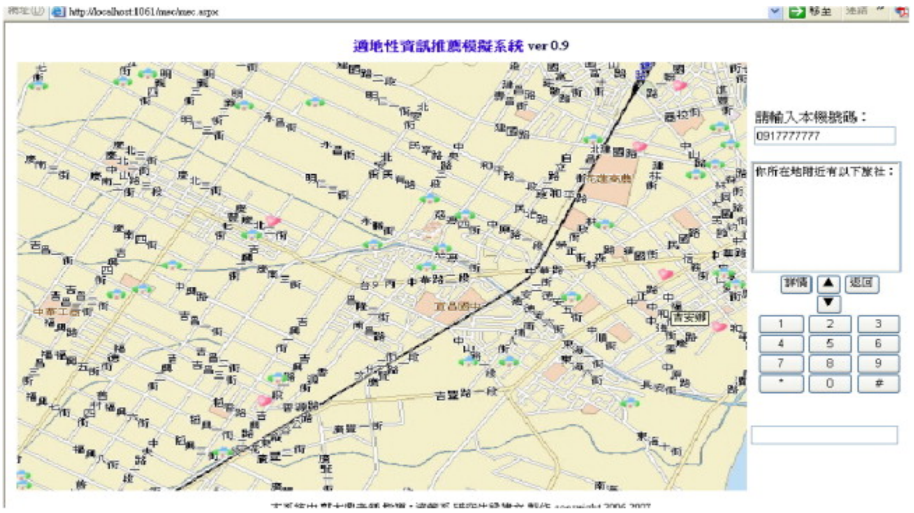

http://localhost:1061/mec/mec.aspx
適地性資訊推薦模擬系統 ver 0.9
詳情
▲
▼
返回
1
2
3
4
5
6
7
8
9
*
0
#
請輸入本機號碼：
0917777777
你所在地附近有以下旅社：

Figure 2.4: Interface in the LBSRM system [33]

2.3 Clustering

Clustering is an efficient tool used to group a set of n-dimensional feature space, into a set of clusters. Clustering algorithms aim to find the maximum amount of similarity in a cluster and to minimize the similarity between the clustered groups[32]. Cluster analysis is defined as "a statistical classification technique for discovering whether the individuals of a population fall into different groups by making quantitative comparisons of multiple characteristics". Broadly, you can divide the clustering algorithms into two main groups: *hierarchical* and *partitional* [31].

In hierarchical clustering algorithms there two approaches. Firstly is an agglomerative method where you start with each data point as a cluster of its own and then merges the similar clusters together to create a hierarchy. The second approach is a divisive top-down approach which starts with all the data points in one big cluster and divides each cluster into two smaller clusters recursively. In partitional clustering, algorithms are used to find all the clusters simultaneously as a partition of the data.

We will utilize a clustering method as a module in our recommendation system as we believe this will allow us to provide better user recommendations. The method will use the user's profile information to match the users information with another set of users who have the same preferences, and most likely will want to visit the same places as you would. There exist a few clustering techniques, and a comparison of their use cases and restrictions is available in the table 2.2.

2.3.1 Data Representation in Clustering Algorithms

The representation of the data is a critical factor that influences the performance of the clustering algorithm. If the representation of the data is optimized, the clusters are more likely to be more compact, and the clusters are more isolated from each other. Unfortunately, there is no universal method to represent the data. A fair strategy is to model the data representation with respect to available domain knowledge. If the representation of the data is not good enough, the clustering method might not find the natural clusters that normally would be present.

2.3.2 Number of Clusters

Deciding on the number of clusters is one of the most difficult problem areas in data clustering. Most of the methods used to find the number of clusters is a problem of model selection. Usually, simulation experiments of clustering algorithms with different n values reveal which value of n is the best one. The value of n is chosen based on problem specific criteria, and which n gives the highest purity in the clusters. The purity of the clusters is used to check if the number of clusters is the optimal number, and that the distances in the clusters are about the same.

2.3.3 K-Means

K-nearest neighbor (k-means) is a widely popular clustering algorithm as it is easy to implement and simple to use. As a consequence of its simplicity, it has its share of drawbacks. Firstly, it does not deal well with overlapping clusters, and the clusters can be pulled out of center by outliers in the data. Secondly, it can get stuck in local minima during execution. Under these circumstances, the resulting clusters are dependent on which of the data points is chosen as the initial seeds. Also, there are data requirements for using k-means as a clustering method. To begin with, you need a populated set of n elements. All of these elements needs to be described using m different attributes, which then is partitioned into K clusters. $X_i = (x_{i1}, x_{i2}, ..., x_{im})$ is used to represent the vector of the m attributes for user i.

Algorithm 2.2 K-Means algorithm

```
Require: n: number of clusters, $X_i$ elements described by m attributes and i:
    iteration limit
1: Select n points at random as cluster centers
2: while i not reached or same points are assigned to each cluster in consecutive rounds
   do
3:     for each object do
4:         Assign the object to their closest cluster center according to the Euclidean
   distance
5:     end for
6:     Calculate the centeroid or mean of all objects in each cluster
7: end while
```

From step 4 in the algorithm 2.2 it uses Euclidean distance the formula for calculating the Euclidean distance in n-dimensions can be seen in equation 2.5.

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_i - q_i)^2 + ... + (p_n - q_n)^2} \quad (2.5)$$

The intention of the K-Means clustering method is to minimize the sum of squared error over all of the clusters found in the data set. Different initializations will most likely lead to disparate cluster groups as K-Means converges to local minima.

Mini Batch K-Means

The Mini Batch K-Means is a version of the K-Means clustering method which improves the time complexity on large datasets [38]. This method utilizes mini-batches to reduce the computation time while still attempting to find an optimized objective function. The mini-batches used are subsets of the data. These subsets are sampled randomly in each iteration. The usage of these mini-batches helps to reduce the amount of computation needed for the method to converge towards a local solution. Compared to other algorithms, which reduce the convergence time of k-means, Mini Batch K-Means produces results that are only slightly worse than the standard algorithm[17].

Mini-batch k-means iterates between two steps much like the standard k-means algorithm. The first step selects samples randomly from the dataset to form a mini-batch, then these batches are assigned to the closest centroid. The second step is to update the centroids with a new position. Algorithm 2.3 shows the pseudocode for the Mini Batch K-Means algorithm.

Algorithm 2.3 Mini Batch K-Means algorithm

```
Require: k, mini-batch size b, iterations t, data set X
 1: Initialize each c ∈ C with an x picked randomly from X
 2: v ← 0
 3: for i=0 to t do
 4:     M ← b examples picked randomly from X
 5:     for x ∈ M do
 6:         d[x] ← f(C, x)                    // Cache the center nearest to x
 7:     end for
 8:     for x ∈ M do
 9:         c ← d[x]                          // Get cached center for this x
10:         v[c] ← v[c] + 1                   // Update per-center counts
11:         $\eta$ ← 1 / v[c]                 // Get per-center learning rate
12:         c ← (1 - $\eta$)c + $\eta$x       // Take gradient step
13:     end for
14: end for
```

Another improved version of the k-means called X-Means can be read about in appendix A.2

2.3.4 Mean shift

Mean shift is a general non-parametric clustering procedure. It takes no assumption on the shape of the distribution nor the number of clusters. The idea behind the mean shift algorithm is to treat the points in the d-dimensional feature space as an empirical probability density function where dense regions in the feature space correspond to the local maxima or modes of the underlying distribution [27]. The algorithm takes advantage of the cluster centroids. The algorithm update the candidates for centroids to be the mean of the points within a cluster. Furthermore, the different candidates go through a post-processing stage to eliminate near duplicates to find the final set of centroids. The algorithm sets the number of clusters, instead of using a parameter which determines the size of the region to search through. This method is not scalable as it requires several nearest neighbor searches during the execution of the algorithm.
Given a candidate centroid x_i for iteration t, the candidate is updated according to Eq. 2.6.

$$x_i^{t+1} = x_i^t + m(x_i^t) \tag{2.6}$$

$N(x_i)$ is the sample neighborhood with a given distance between x_i and m, also known as the mean shift vector. This vector is computed for each centroid that points towards a region of the maximum increase in the density of points [17]. The $m(x_i)$ is in fig. 2.6 is computed with the equation shown in Eq. 2.7

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i)x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)} \tag{2.7}$$

Mean shift clustering utilizes bandwidth selection. There are four different techniques used to find the bandwidth.

- *Statistical method:* This method finds the optimal bandwidth associated with the kernel density estimator; this estimator is defined as the bandwidth that achieves the best compromise between the bias and variance of the estimator overall for $x \in R^d$ [26].
- *Stability of decomposition:* The bandwidth is the center of the larger operating range over which the same number of clusters are obtained.
- *Objective function:* The best bandwidth maximizes an objective function that expresses the quality of the decomposition. The objective function typically compares the inter- vs. intra-cluster or evaluates the isolation and connectivity of clusters.
- *Top-down:* Top-down information provided by the user or by an upper-level module can be used to control the kernel bandwidth.

A more efficient mean shift clustering method requires a resampling of the input data with a regular grid. This technique is used in the context of density estimation which leads to a binned estimator. Further, the reduction in the computation time is achieved by employing algorithms for multidimensional range searching and used to find data points falling in the neighborhood of a given kernel [26].

2.3.5 BIRCH

Balanced iterative reducing and clustering using hierarchies (BIRCH) incrementally and dynamically clusters the incoming multi-dimensional data points to produce the best possible clusters in the data. The BIRCH clustering method demonstrate that it is especially suitable for substantially large databases. Its I/O cost is linear in the size of the data set: a single scan of the dataset yields a good clustering, and one or more additional passes can be used to improve the cluster quality[56]. There are a few advantages to the BIRCH approach compared to other clustering methods.

- BIRCH is a local method, meaning that the cluster decisions is made without scanning all the data points currently existing in the clusters. Instead, it uses measurements that reflect the closeness of points, and at the same time being incremental during the clustering process.
- BIRCH uses observations that the data space is usually not uniformly occupied, hence not every data point is equally important for clustering purposes. Furthermore, BIRCH sees dense regions of data points as one single cluster, and points in sparse regions are treated as outliers and can be removed.
- BIRCH clustering and reduction process is organized and characterized by the use of in-memory, height-balanced and highly-occupied tree structure.

$$\begin{aligned}\overrightarrow{x_0} &= \frac{\sum_{i=1}^{N}\overrightarrow{x_i}}{N}\\ R &= \left(\frac{\sum_{i=1}^{N}(\overrightarrow{x_i}-\overrightarrow{x_0})^2}{N}\right)^{\frac{1}{2}}\\ D &= \left(\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(\overrightarrow{x_i}-\overrightarrow{x_j})^2}{(N(N-1)}\right)^{\frac{1}{2}}\end{aligned} \tag{2.8}$$

BIRCH terminology of vector spaces $i = 1, ..., N$ centroid $\overrightarrow{x_0}$, radius R and diameter D for figure 2.8. R is the average distance from cluster members to the centroid. D is the average pairwise distance within a cluster. These two, R and D, are alternative measures of how tight the clusters are around the cluster centroid. The concept of clustering feature and CF tree are the core elements in BIRCH incremental clustering [56]. The CF tree is a height-balanced tree which has two different parameters: branching factor B and threshold T. Each of the leaf node contains at most B entries, and each of the nodes represents a cluster made up off all the subclusters represented by its entries. All of the entries must satisfy a threshold requirement which threshold radius and the entries needs to be less than T. The CF tree is a way to view the data in a compact representation. Each of the entries in a leaf node is not a single data point, but a sub-cluster.

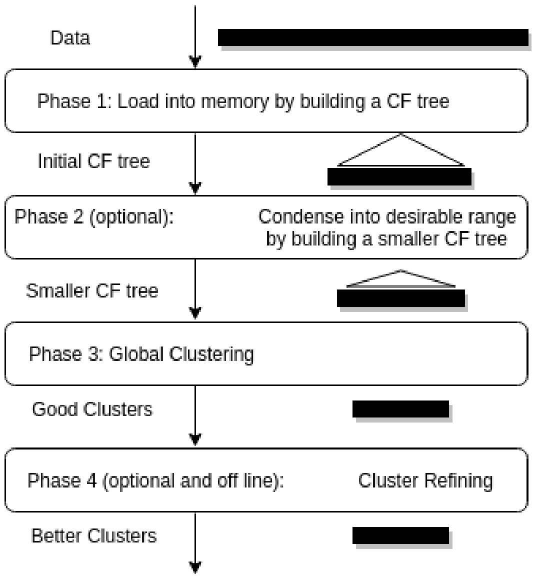

Data
Phase 1: Load into memory by building a CF tree
Initial CF tree
Phase 2 (optional):
Condense into desirable range by building a smaller CF tree
Smaller CF tree
Phase 3: Global Clustering
Good Clusters
Phase 4 (optional and off line):
Cluster Refining
Better Clusters

Figure 2.5: Overview of BIRCH [56]

The figure 2.5 shows an overall concept of how BIRCH clusters the data points. Phase 1 is to scan the data and build a CF tree using the available memory and space on disk. The CF tree in phase 1 tries to reflect the information in the data set as good as possible under the limit of the memory. The data is grouped as fine subclusters, and sparse data points are removed as outliers. Phase 2 is an optional phase in the BIRCH algorithm. This phase is supposed to provide a cushion between phase 1 and 3; this is done because different clustering methods used in Phase 3 have different input size ranges and phase 2 tries to bridge this gap. After phase 3, we get a set of clusters that captures the major distribution pattern in the data, but inaccuracies can exist due to of a misplacement problem with BIRCH. Phase 4, which is also optional, uses the centroids of the clusters produced in earlier phases as seeds and redistributes the data points to its closest seed to get a new set of clusters.

Experimental data tells us that BIRCH performs very well on several large datasets, and is superior regarding quality speed and order-sensitivity to other clustering algorithms previously used on large datasets. The settings of the parameters to a proper setting is necessary for the efficiency of the BIRCH algorithm, but BIRCH does not scale very well with high dimensional data. If the number of features is greater than twenty, Mini Batch K-Means 2.3.3 will yield better results[17].

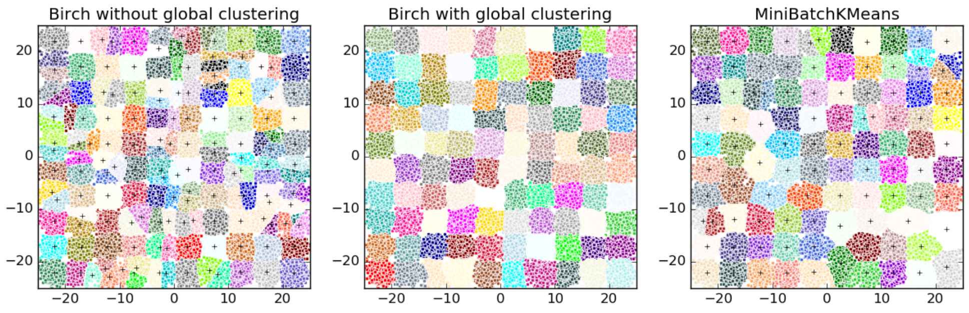


Figure 2.6: Non-global and global BIRCH clustering vs Mini Batch K-Means clustering[1]

[1]http://scikit-learn.org/stable/modules/clustering.html#birch, accessed 25.05.2016

Method name	Parameters	Scalability	Usecase	Geometry(metric used)
K-Means	Number of clusters	Very large number of samples, medium number of clusters	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with large samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	Bandwidth	Not scalable with number of samples	Many clusters, uneven cluster size, non-flat geometry	Distance between points
Spectral clustering	Number of clusters	Medium number of samples, small number of clusters	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large number of samples and number of clusters	Many clusters, possibly connectivity constraints	Distance between points
Agglomerative clustering	Number of clusters, linkage type and distance	Large number of samples and number of clusters	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	Neighborhood size	Very large number of samples, medium number of clusters	Non-flat geometry uneven cluster sizes	Distances between nearest points
Gaussian mixtures	Many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	Branching factor, threshold, optional global cluster	Large number of clusters and number of samples	Large dataset, outlier removal, data reduction	Euclidean distance between points

Table 2.2: A comparison of clustering algorithms [17]

2.4 Dimensionality Reduction and Visualization

Dimensionality reduction is important in many domains since it mitigates the curse of dimensionality and other undesired properties of high-dimensional spaces. The last decade it has been introduced a significant number of nonlinear techniques to deal with complex nonlinear data. These complex nonlinear data are often real-world data [37]. As shown in figure 2.7, it is common to divide the techniques into convex and non-convex. The convex methods optimize the objective function used and do not contain local optima, and the non-convex methods optimize the objective function which holds local optima. See Appendix A.3.1 for more information about these techniques. From the results obtained by “Dimensionality Reduction: A Comparative Review” [37], the conclusion is that the nonlinear techniques are often not capable of outperforming the traditional linear techniques.

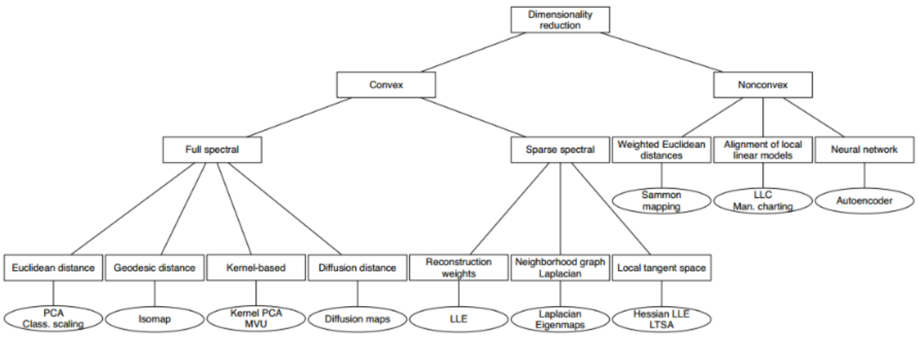


Figure 2.7: Taxonomy of dimensionality reduction techniques [37]

There exist some nonlinear dimensionality reduction techniques that aim to preserve the local structure of the data, some of the techniques are.

1. Sammon mapping
2. Curvilinear Components Analysis (CCA)
3. Stochastic Neighbor Embedding (SNE)
4. Isomap
5. Maximum Variance Unfolding (MVU)
6. Locally Linear Embedding (LLE)
7. Laplacian Eigenmaps

Even with the strong performance you get from these methods on artificial data sets, they are often not very successful at visualizing real, high-dimensional data. Most of the techniques are not capable of retaining both the local and the global structure of the data in a single map. For instance, some of the techniques are not capable of separating handwritten digits into their natural clusters [36].

2.4.1 SNE

Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between the data points into conditional probabilities which represent the data points similarities. The similarity between two data points is given with the conditional probability $p_{i|j}$. The conditional probability will be high for close data points, and infinite small for widely separated data points. σ_i is the variance of the Gaussian that is centered on data point x_i.

$$p_{i|j} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)} \tag{2.9}$$

The mapping of the low dimensional data is y_i and y_j and it makes it possible to compute a similar conditional probability $q_{i|j}$.

$$q_{i|j} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2)} \tag{2.10}$$

If the mapping of the points y_i and y_j is correctly modeled, they will be similar to the high-dimensional data points x_i and x_j and the conditional probability $p_{i|j}$ and $q_{i|j}$ will be similar. SNE uses this as a motivation to find a low-dimensional representation of the data which minimizes the mismatch between $p_{i|j}$ and $q_{i|j}$. SNE minimizes the sum of Kullback-Leibler divergences over all data points using gradient descent method. The cost function C is given:

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{2.11}$$

SNE tends to find maps with a better global organization. Unfortunately, this requires sensible choices of the initial amount of Gaussian noise and the rate at which it decays. Moreover, these options interact with the amount of momentum and the step size that are employed in the gradient descent. And therefore, there can be useful to run several optimization's on the data set to find the best parameters [36].

2.4.2 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction method capable of capturing most of the local structure of the high-dimensional data very well while still reveals the global structure such as the presence of clusters at several scales [36]. The t-SNE technique uses SNE to reduce the dimensionality of the data. SNE constructs reasonably good visualizations. However, it's weakness is its cost function which is difficult to optimize and is referred to as the "crowding problem". The technique t-SNE aims to alleviate these problems. It uses another cost function in two different ways. Firstly, it is that it uses a symmetric version of the SNE cost function with simpler gradients. Secondly, it uses of Student-t distribution rather than a Gaussian to compute the similarity between two points in the low-dimensional space.

It is possible to minimize a single Kullback-Leibler divergence between a joint probability distribution P in the high-dimensionality space, and a joint probability distribution Q in the low-dimensionality space:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2.12}$$

This is refereed to as symmetric SNE. It has the property that $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$ for $\forall_{i,j}$. In symmetric SNE, the pairwise similarities in the low-dimensional map q_{ij} are given by:

$$q_{ij} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq l} exp(-||y_k - y_l||^2)} \tag{2.13}$$

For the high-dimensional space p_{ij}:

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_k - x_l||^2/2\sigma^2)} \tag{2.14}$$

Equation 2.14 causes problems when the data point x_i is an outlier. In this case, the value of p_{ij} will be extremely small for all j. The problem is then with the location of its low-dimensional map point y_i has little to no effect on the cost function used in symmetric SNE. The problem is circumvented by defining the joint probability in the high-dimensional space to be the symmetric conditional probability $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$. The result of using this definition in the symmetric SNE is that each point x_i makes a contribution to the cost function.

Symmetric SNE is matching the joint probabilities in pairs of data points in the high space and the low rather than the distances. It has a natural way of avoiding the crowding problem because the high space we convert the distances into probabilities with a Gaussian distribution. Meanwhile in the low space, it uses a probability distribution with heavier tails than the Gaussian to calculate the distance into probabilities. The t-SNE uses a Student t-distribution with one degree of freedom as a heavy-tailed distribution in the low dimensional map [36]. Due to the dainty properties of the Student t-distribution, it makes the map representation from a high-space to a low space almost invariant to changes. This means that clusters interact in the same way with each other in the low space as in the high space.

Algorithm 2.4 Simple version of t-Distributed Stochastic Neighbor Embedding

```
Require: data set X = $x_1, x_2, ..., x_n$, cost function parameters, perplexity Perp,
    optimization parameters: number of iterations T, learning rate $\eta$, momentum
    $\alpha$(t)
 1: Result: low-dimensional data representation $\Upsilon^T = y_1, y_2, ..., y_n$
 2: Begin
 3: compute pairwise affinities $p_{j|i}$ with perplexity Perp using equation 2.9
 4: set $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$
 5: sample initial solution $\Upsilon^0 = y_1, y_2, ..., y_n$ from $N(0, 10^{-4}I)$
 6: for t=1 to T do
 7:     compute low-dimensional affinities $q_{ij}$
 8:     compute gradient $\frac{\delta C}{\delta \Upsilon}$
 9:     set $\Upsilon^t = \Upsilon^{t-1} + \eta\frac{\delta C}{\delta \Upsilon} + \alpha(t)(\Upsilon^{t-1} - \Upsilon^{t-2})$
10: end for
11: End
```

A comparison of the result of the visualization techniques reveals the strong performance of t-SNE compared to the other techniques you can see listed in 2.4 [36].

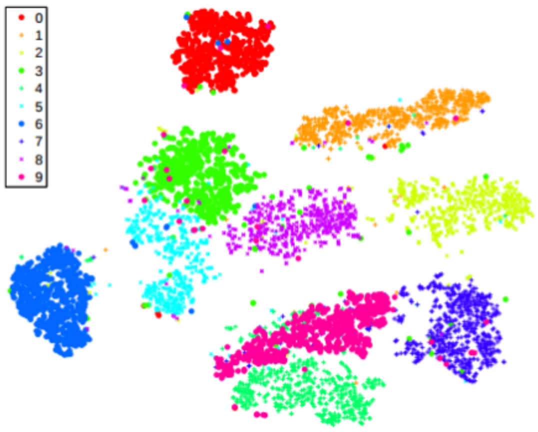


Figure 2.8: Visualization by t-SNE of the MNIST data set [36]

For figures of other visualization techniques using the MNIST data set on 6000 digits, see Appendix A.3.2. Were the different techniques are used on different data sets used in “Visualization of Data using t-SNE” [36], we can see that the overall performance of t-SNE is better than the previously used visualization techniques. Although t-SNE performs well compared to other techniques for data visualization, there are three potential weaknesses.

1. It is unclear how t-SNE performs on general dimensionality reduction tasks
2. The relatively local nature of t-SNE makes it sensitive to the curse of the intrinsic dimensionality of the data
3. t-SNE is not guaranteed to converge to a global optimum of its cost function

Chapter 3 Approach

In the following chapter, we will discuss the systems requirements needed for the system to function properly. We look closer at the different methods we thought promising, and what conditions needs to be satisfied before moving forward. We will also mention the design decisions we have taken, and what we require from the user for the system to work as intended.

3.1 Requirements

For the system to work correctly, some requirements need to be met. Among others, there are some preconditions the users will need to meet before they can use the system, and both the client and server need to fulfill the system requirements before the system can give results back to the user.

3.1.1 Functional Requirements

The functional requirements have each been given a priority rating in the scale High-Medium-Low. High priority is most prioritized and essential for the system to perform at the required level of the service. Medium prioritized requirements are not necessary for the system to run, but they enhance the quality of the system and will yield a better user experience when using the service. Low priority requirements are non-essential requirements but are features which are nice to have, and likely to be incorporated in later versions of the system. Table 3.1 lists all the functional requirements for our recommendation system.

ID	**Name**	**Description**	**Priority**
FR 1	User profile creation	The user should be able to create a profile in the system.	High
FR 1.2	User profile update	The user should be able to update their profile and their preferences in the different categories.	High
FR 1.3	User profile deletion	The user should be able to delete their profile.	Medium
FR 1.4	Users radius of search	The user can change the radius of their attraction search.	Low
FR 2	No duplicated usernames	The users cannot have duplicated usernames.	High
FR 3	User login	The user should be able to login using the username and password created in FR 1.	High
FR 3.1	User logout	The user can log out of the system.	Medium
FR 4	Search for attractions	The user should be able to search for attractions in the system.	High
FR 4.1	Mark attraction as visited	The user will be able to mark an attraction as visited.	High
FR 4.2	Mark attraction as liked	The user will be able to mark an attraction as liked.	High
FR 5	Recommend attractions	Recommend attractions similar users have liked or have visited.	High

Table 3.1: Functional requirements for the system

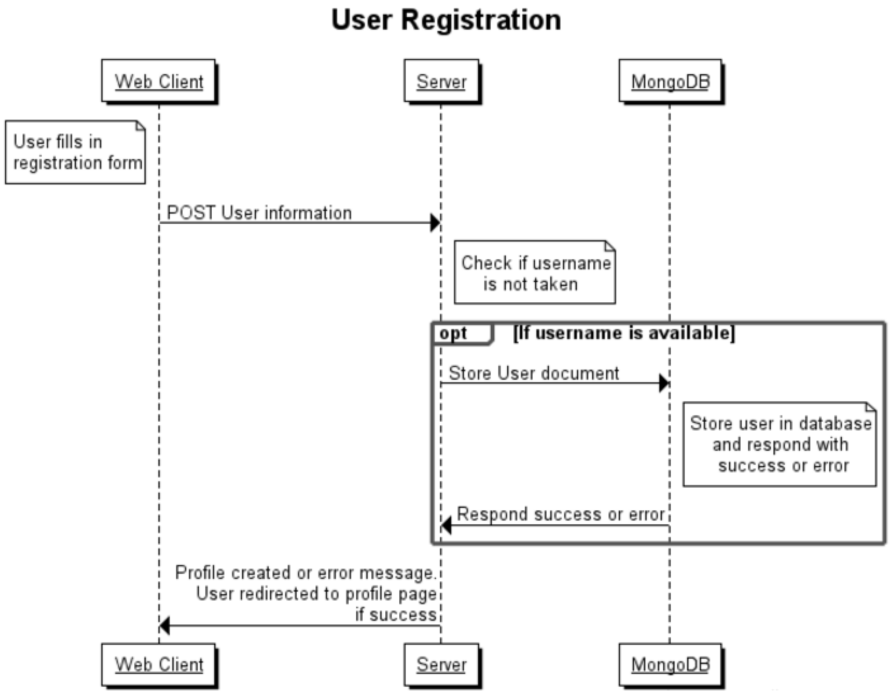

User Registration
Web Client
Server
MongoDB
User fills in registration form
POST User information
Check if username is not taken
opt
[If username is available]
Store User document
Store user in database and respond with success or error
Respond success or error
Profile created or error message. User redirected to profile page if success
Web Client
Server
MongoDB

Figure 3.1: A sequence diagram for user registration

3.1.2 System Requirements

The recommendation component itself is the most important part of our service since it is responsible for finding suitable activities and attractions for the users. The system does so by considering the user profile and user history to cluster users together and give recommendations based on other users' visits and likes. Here is a brief description of system requirements from the system utilizing user-user CF to provide recommendations to the users.

Figure 3.1 shows the interaction between the user and the recommendation system during a registration process. More sequence diagrams are available in Appendix B.3.

Geographical Search

The system will be geographical aware meaning the user searches for activities yield results within a predefined radius either around the location of the user or a user's defined search area. The range of the search can not be adjusted by the user to include a larger or smaller area at the time, but is likely to be included in later versions of the system. For the system to be geographical aware, we need a solution to getting the users location. We have delegated this requirement to the web client.

Implicit Feedback From User

We want feedback in our recommendation system to be easy, fun and straightforward. The user should not feel that the feedback process holds them back in their interaction with the service. Therefore, we have chosen implicit feedback as our way to gather feedback from the users; the implicit feedback collected from the users are done by letting them chose if they want to visit an attraction or if they have visited a specific attraction. This method is almost an explicit feedback method since the user has to click specific buttons to leave feedback, but it is classified as implicit feedback since it is a fast and easy way for the user to register their interests in the system.

Clustering of users

For our vision of creating a user-user CF recommendation system, we will need to find users who share interests and are closely related to each other. Therefore, we will need to cluster the users actively based on both their user profile preferences and their user history to get more precise groups. Our recommendation system has to consider various categories when providing recommendations. Food taste does not have an effect on your taste in music, not in an easily measurable way. Consequently, we decided to cluster users for each category individually as we believe that will result in the most optimal way to finding similar users like yourself which naturally will lead to a better-suited basis for our recommendations.

$$\begin{aligned} \texttt{Profile vector category i} = \\ = \texttt{preference category i} + (0.25 * numberofvisitstocategoryi) \end{aligned} \tag{3.1}$$

History Awareness

The recommendation system will be history aware. For our purpose, it means that the system will gather information about the users' activities and use this information to help recommend future activities and give more precise recommendations. Therefore, the system will need to accumulate information from the user and store it in a database such that the data easily can available to support in the recommendation process.

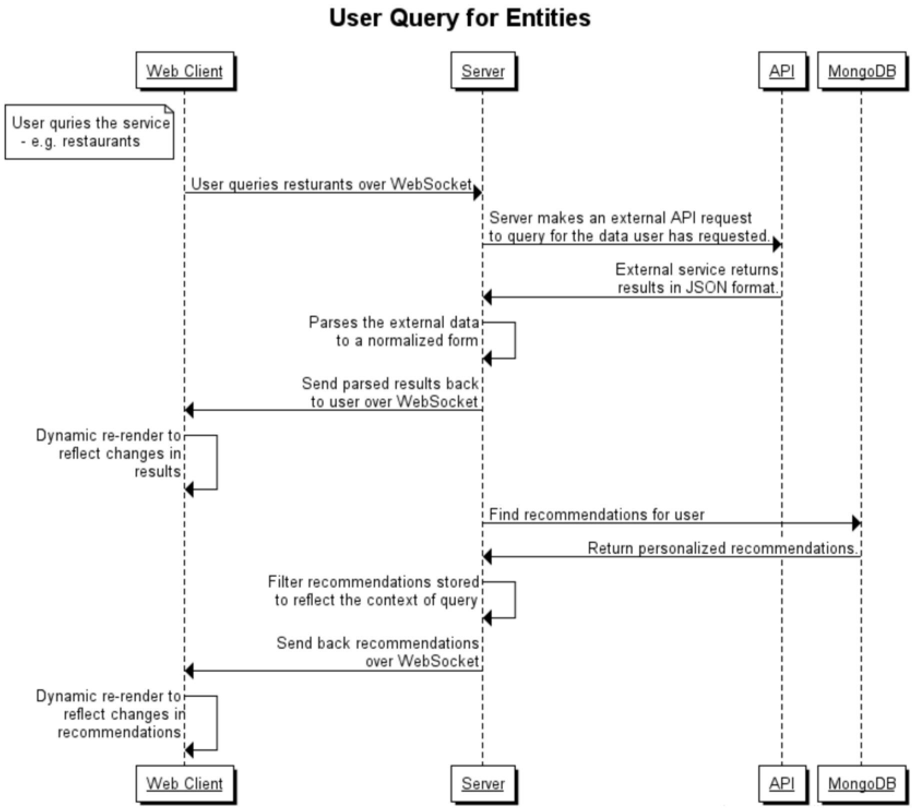

User Query for Entities
Web Client
Server
API
MongoDB
User quries the service - e.g. restaurants
User queries resturants over WebSocket
Server makes an external API request to query for the data user has requested.
External service returns results in JSON format.
Parses the external data to a normalized form
Send parsed results back to user over WebSocket
Dynamic re-render to reflect changes in results
Find recommendations for user
Return personalized recommendations.
Filter recommendations stored to reflect the context of query
Send back recommendations over WebSocket
Dynamic re-render to reflect changes in recommendations
Web Client
Server
API
MongoDB

Figure 3.2: A sequence diagram for when user query the our service

Client

The client in our service is a web application and should be able to support the following listed features. These features are essential for the service to function properly and give the user what is expected from the system.

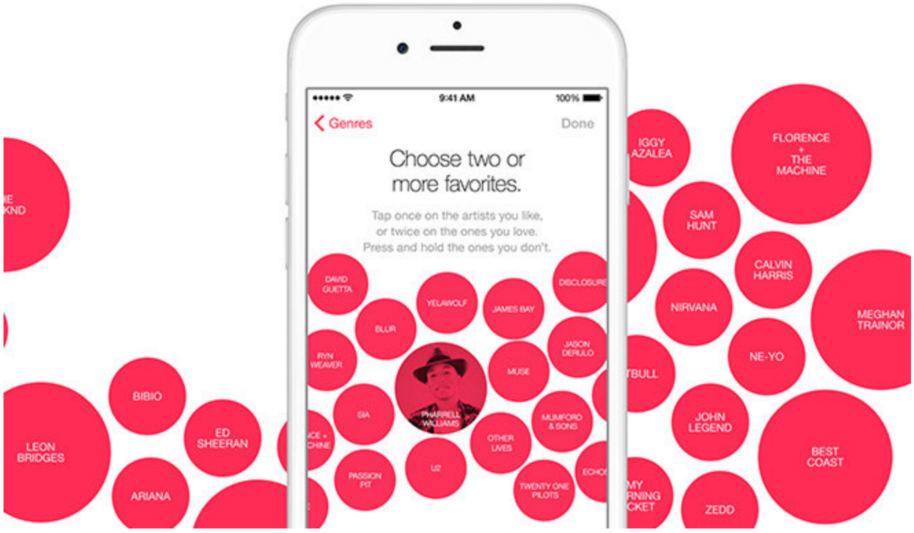
`Profile Creation`
The client has to have a way for the user to create a user profile in our system. The registration process should be an easy and clean process for the user to complete without any help. When a user creates a profile, they will be asked to fill in a username, password, their age group, gender, country and state of residence. If the username already exists, the user should get an error message. Furthermore, when the profile creation is complete, the user will immediately be taken to his or hers profile page and asked to update their preferences. To help the user understand how to use the service we believe a help section will suffice for the web application. See Figure 3.9 for the mockup of the registration page.

`Profile Page`
The service client should not only be able to create a user profile, but it will also have to feature a profile page to get more information about the user into the service. We want the user to tell us what activities they like, dislike and are neutral too. The limited number of choices are since we have a small number of users in the system to start with, and therefore it would be hard to give good recommendations to the users if we had a span from 0-9 or 0-99 of how much the user like one activity. Furthermore, we ask the user to fill in individual characteristics such as age group, gender, country, and state of residence. Finally, the user will be able to review previous heart or starred activities or attractions, and remove them should the user no longer like them in his or hers profile history. We will get back to the intention of heart and stars from the users. Figure 3.11 shows the mockup for the profile page.

This approach to creating a service is like what Apple did it when they launched their service Apple Music on June 30, 2015. They asked the user to create a profile and to select two or more of their favorite music categories and artists. Requesting for knowledge from the user is done to prevent the problems affiliated with the "cold start" problem and give the users good recommendations from the beginning of the service. See Figure 3.3 for an example of their interface.

`Filtering and Distance of Search`
The client will be able to get the location of the user and thereby filter the search area. As such, the service only will recommend attractions which are within a certain distance of the user. Ideally, the user would be able to change the scope of the search by adjusting a search parameter. There is a huge difference whether the user is on foot in a high-density city or has access to a car in a low-density area, but we have decided not to accommodate this problem at this time, but is likely to be included in an update to the service at a later time.

We do not require the user to be upfront about their location. Consequently, if the user is unwilling to allow the web application to retrieve geolocation data, the user will have to specify their location manually to for the recommendation system to yield results and recommendations. The same fallback mechanism is used then the user wants to explore other areas than currently located.


Choose two or more favorites.

Tap once on the artists you like, or twice on the ones you love. Press and hold the ones you don't.

Figure 3.3: Apple music user profile creation interface[1]

Server

The requirements for our server to function and process the data required, and for the system to serve recommendations to its users.

`Clustering`

We aim to use the users and their behavior as the backbone when providing recommendations. We will need to cluster the users into groups where each group contains users who have the same taste and interests. We created an anonymous survey using Google survey [29] to gather real life user profiles. Both to test the clustering methods and try to find the ideal number of clusters from a given number of users. Google survey made it easy to collect data in a matter of minutes; the answers is also continuously updated after every answer; you can easily get different graphs from the survey that displays various parts of the responses. See figure 3.5 and 3.6. The questionnaire asked for the same user data as what we ask for upon account creation. A complete overview of information we asked the participants to fill is available in Appendix B.1.1. An important factor to remember when clustering users, and using a user-user CF to give recommendations, is to have the clusters not to be too narrow or too broad. If the groups are too small, it will not contain enough diversity to give good recommendations the users like and might not have considered. However, if they are too large, you will have a too broad spectrum of items to choose from and

[1]Figure taken from *http://appleinsider.com/articles/15/06/10/everything-you-need-to-know-about-apple-music* on 13.05.2016

the system then might consider recommending entities that are completely out of the question.

We tested different parameters of clustering both with and without demographic data and a different amount of user profiles. We did this to try to find out how many clusters which were optimal given a known number of users. We used WEKA and the user profiles we got from the survey answers when running our cluster simulations. WEKA is a collection of machine learning algorithms for data mining tasks[55]. The algorithms WEKA utilizes can be applied directly to the dataset. WEKA provides clustering tools like simple K-Means (K-Nearest neighbor), DBSCAN and Hierarchical clustering. In our tests, we used simple K-Means to cluster the user profiles from the survey. The results can be seen in table 3.2, 3.3, and in the Appendix B.1.4. From WEKA, we extracted the information: the sum of squared errors, the number of iterations used to find clusters, the time used to cluster users and the minimum and the maximum number of users in the clusters created. WEKA also requires a certain format on the data set as you can see this in Appendix B.1.4.

Due to a potentially large user group in a tourist attraction system, the clustering method needed in the system will have to handle a large amount of data in the five different categories. The method chosen also needs to be fast and give good and consistent results. With these requirements in mind, we decided to look closer at the clustering methods K-Means, Mini Batch K-Means and Birch. We believe all of these methods meets our clustering needs for the recommendation system. You can see different clustering methods in table 2.2.

K-Means and Mini Batch K-Means is almost the equivalent algorithm although the Mini Batch K-Means gives better runtime on a large dataset. The resulting clusters small noticeable differences as seen in figure 3.4. One of the downsides of using the K-Means and Mini Batch K-Means is that you will need to set the number of output clusters as a parameter in the algorithm. Therefore, we will need to find an equation which confidently will give the best number of clusters from the total number of users. Results from simulation runs we did with the data gathered from the survey can be seen in figure 3.7 and 3.8. We found that the equation 3.2 would scale the number of clusters approximately to ideal number of clusters as the userbase increases.

The BIRCH method is another clustering technique we expect can be a useful clustering method to be used in the system. The BIRCH method handles large datasets, consider outliers in the data and it finds the number of clusters which is optimal for the data. It is slower than the K-Means and Mini Batch K-Means algorithms, and it might require for the dataset to run twice to find the best clusters. After careful consideration, we have chosen to go forward with using the K-Means method as our initial clustering algorithm. Should the number of users using the system reach a higher level than anticipated and the clusters from K-Means does not give satisfactory results, the change from K-Means to BIRCH can be accomplished without to much additional work on the server side.

$$number\ of\ clusters = \lceil \log_2(number\ of\ users) \rceil \tag{3.2}$$

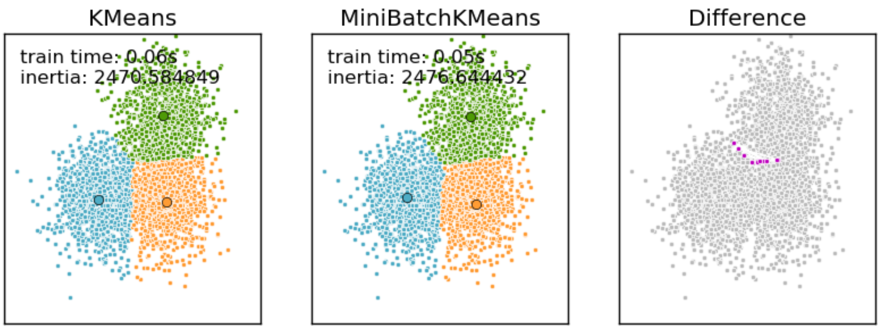


Figure 3.4: A comparison of clusters between the K-Means and Mini Batch K-Means methods[2]

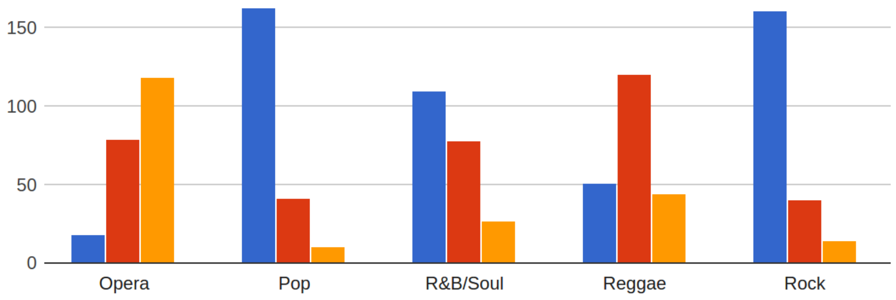


Figure 3.5: Distribution of preference in music genres from survey[3]

[2]http://scikit-learn.org/stable/modules/clustering.html#mini-batch-k-means accessed 30.05.2016

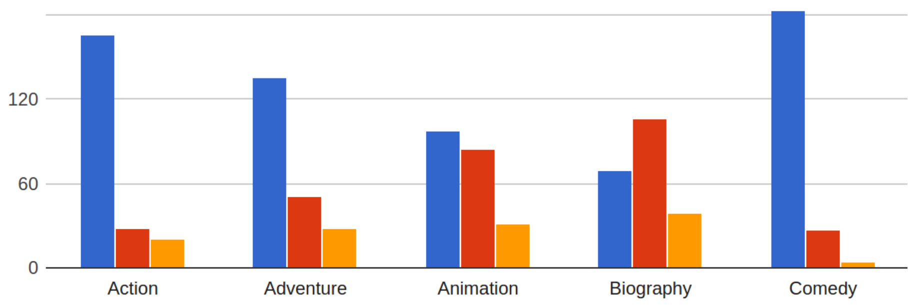


Figure 3.6: Distribution of preference in movie genres from survey[3]

[3]Blue means like, red means neutral and orange means dislike

Number of clusters	**Sum of squared errors**	**Iterations used**	**Time to build model [sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	2042	6	0.08	79	137
3	1951	3	0.03	45	88
4	1856	5	0.01	43	79
5	1808	6	0.02	15	81
6	1725	6	0.02	16	54
7	1657	5	0.01	18	48
8	1635	5	0.04	14	45
9	1598	7	0.03	11	38
10	1583	5	0.01	5	40
11	1555	8	0.03	5	37
12	1535	6	0.02	6	35
13	1476	9	0.02	6	33
14	1451	6	0.1	4	27
15	1430	7	0.02	6	30
16	1411	7	0.02	3	30
17	1398	7	0.01	3	30
18	1383	6	0.01	3	26
19	1352	7	0.01	4	24
20	1321	8	0.02	4	24
25	1281	5	0.02	3	25
30	1211	8	0.04	1	19
40	1112	7	0.02	1	15

Table 3.2: Simple K-Means on Movies with sex, age and marital-status using Euclidean distance, max iterations 500, 10 seeds using WEKA [55]

Number of clusters	**Sum of squared errors**	**Iterations used**	**Time to build model [sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	1780	6	0	79	137
3	1643	7	0	46	105
4	1520	6	0	22	85
5	1460	8	0.01	20	79
6	1414	7	0.01	16	74
7	1403	7	0.01	8	53
8	1392	5	0.02	9	41
9	1354	7	0.01	9	40
10	1342	6	0.01	5	39
11	1309	7	0.01	7	40
12	1291	9	0.01	4	45
13	1253	10	0.01	4	38
14	1232	7	0.01	4	34
15	1199	9	0.01	3	37
16	1196	9	0.02	2	37
17	1198	5	0.01	2	33
18	1176	6	0.01	2	34
19	1141	9	0.02	2	31
20	1113	9	0.01	3	32
25	1045	7	0.01	3	32
30	1017	6	0.02	1	29
40	893	11	0.03	1	13

Table 3.3: Simple K-Means on Movies without sex, age and marital-status using Euclidean distance, max iterations 500, 10 seeds using WEKA [55]

For more clustering tests, see Appendix B.1.4.

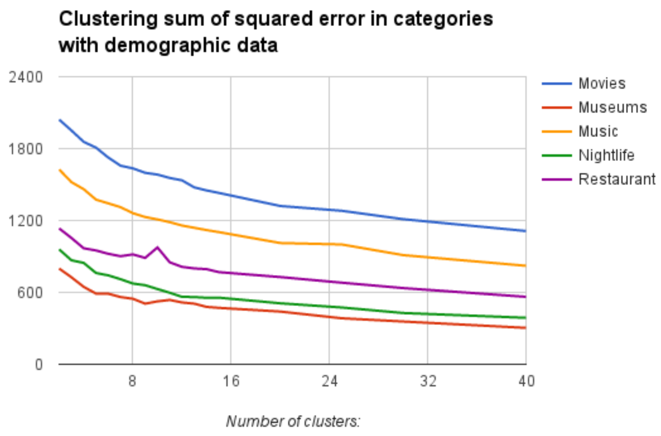


Figure 3.7: The sum of squared error in the different categories with demographic data using K-Means

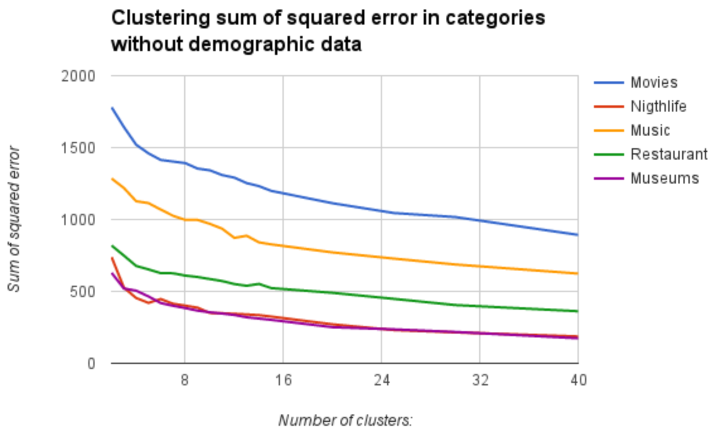


Figure 3.8: The sum of squared error in the different categories without demographic data using K-Means

3.2 User Requirements

There are some requirements needed from the user before he or she can use our recommendation system. Here is a list of these, and the reason behind them.

3.2.1 Profile Requirements

For the user to be able to to use the tourist attraction recommendation system, the user will need to create a user profile. This user profile contains information about the users and their preferences in the different categories. As already mentioned in section 3.1.2, this user data is used to cluster related users taste wise to provide excellent recommendations. Also, the users history are considered in the recommendation module.

Not only are we asking the user to create a profile before using the system, but the user is also encouraged to fill in his or her preferences upon account creation. The reasoning behind this is to avoid the typical "cold start" problems associated with recommendation systems. The quicker we learn our users likes, and dislikes, the better the recommendations will become. Especially early on in the lifespan of the service, the recommendations are susceptible to poor quality as the service has insufficient information to work around. The ask the user to fill in their preferences within five domain categories and their respective subcategories, the very same areas the recommendation system will provide recommendations. These domains are the following:

- Movies
- Museums
- Music
- Nightlife
- Restaurants

To see the categories, subcategories and what personal information the user can fill in when creating or updating the user profile, see Appendix B.1.1. Due to a most likely a spare number of users in the system, the preference ratings had to be simplistic. We have decided to let category preferences stored in the user database as an integer indicating whether or not they like(1), dislike(-1) or are neutral(0) to said category.

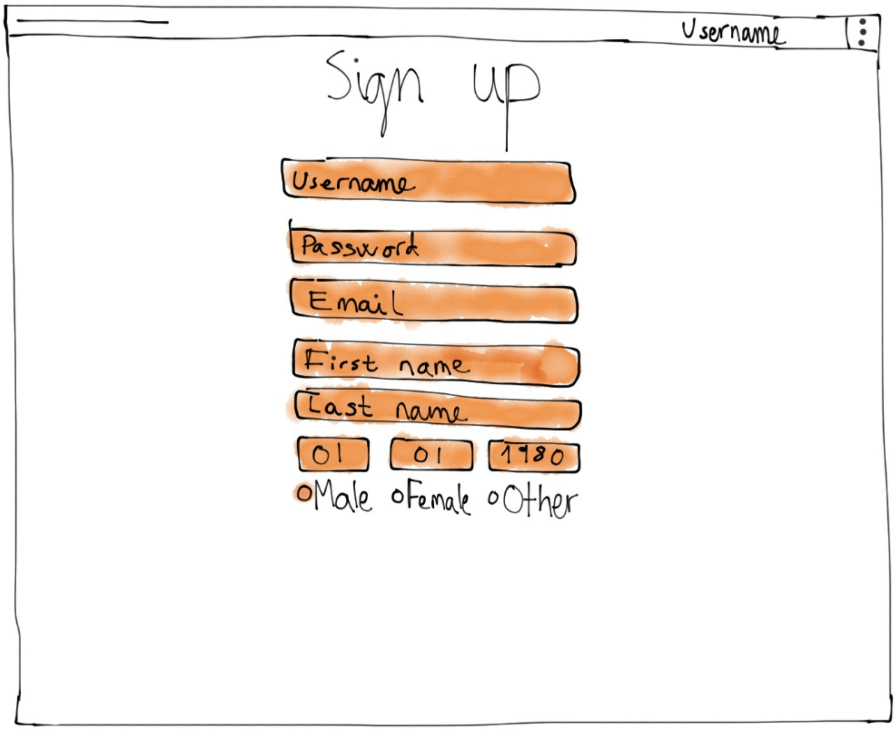


Figure 3.9: Proposed user registration page

It is required that the user can see and change their preferences in their user profile page. The users will also see the list of the attractions they have liked and the list of attractions they have visited. If a user has visited a tourist attraction or activity, the system will show it as a starred attraction. The user will heart attractions they would like to visit. See figure 3.9 and 3.11 for proposed user interface. The system will learn more about the user over time as the user spend time marking entities as hearts or stars. The recommendation system will have an easier time finding similar users and providing user-user recommendations with this approach.

3.2.2 Modern Browser

For the user to be able to us the service we require the user to have a modern browser to be able to view the content of the service. Many of the key technologies used in the project depend on an adequately up-to-date browser.

3.2.3 Geolocation

We require the system to be able to collect the geolocation from the user, but the user can opt not to give us access to their location. The recommendation system will take advantage of knowing the users' locations. The geolocation collected from the user's device will be used to set the users location, and it will be used to find activities and attractions in the area near the users' location. However, if the user does not wish to give up their location, they will have the possibility to set their location manually by moving the focus area for the applications map to their destination and find suitable activities and attractions in the new area.

3.3 API Requirements

For the external database APIs, we require datasets which contain information about the five categories and their subcategories such that the system will be able to give a recommendation to a user in all of the categories. To our knowledge, there do not exist a single external API that provides us with all the information that we need, which is why we need to implement and support multiple sources of external data from various APIs.

3.3.1 Dataset Maintenance

A key aspect of our service is that it will be working with real-time data from both the user and from the database. Consequently, we require an up-to-date and well-maintained database to strengthen our service. It is critical to avoid recommending businesses that no longer exist since such recommendations reflect poorly on our service, and might quickly lead to a diminishing user base.

3.3.2 Metadata in Dataset

Our service needs to be able to sort the entities and differentiate them from one another. An example of this is that a restaurant is not just a restaurant. It might be a steakhouse or an Indian restaurant, and these serve different cuisines which might not be everyone's preference. Accordingly, we require the metadata in the API is well defined and contains the correct metadata for all of the categories.

3.3.3 Possible Datasets

Considering our categories, a well-maintained external source of information, and provided utility and metadata, the APIs we took a closer look at was TripAdvisor, Google Places, Foursquare, Songkick, and Yelp.

For our source of upcoming concerts and artists appearances, we found the third-party service Songkick[18] to suit our needs. Songkick offered extensive and high-quality content, which we immediately recognized. We are confident moving forward that Songkick is the right fit. As for restaurants, museums and nightlife, we took a look at TripAdvisor, Google Places, Foursquare, and Yelp. Tripadvisor[20] did not allow access to their content API for the purpose of academic research it was the first to be cut from our list as a possible source.

Yelp[22] was a strong contender to our primary source of data. However, after a closer look and revealing questionable data quality of the data around the world, we left Yelp out of our list. Yelp would be a perfect match for users from the US, but in Norway Yelp would be far from optimal. With Google Places and Foursquare left, and our primary choice ended up being Foursquare since we believed the user terms of the API at Foursquare[4] is better than Google Places[5], and the maintenance the data in Foursquare was as good or almost as good as Google Places. For providing movie showings, we looked into collecting data from Google Showtime[6] via scraping data of their service. A quick glimpse reveals that Google Showtime has its flaws, but to our knowledge, there exist no other solution except perhaps going movie theaters' web pages directly.

3.4 Design

The design of our application aims to be easy to use for both youths and seniors. We use Bootstrap[21] to quickly develop a responsive web application, which not only looks good on multiple platforms but is also feature stacked. For now, the priority is to create a web application service, and Bootstrap eases the development process both for computer and mobile platforms. Nonetheless, native apps are the ideal direction for providing convenience for the mobile users, but Bootstrap became the perfect compromise now in the initial stages of the development.

3.4.1 Web Application

The web application is the only way to use the tourist recommendation system we have developed. We also discussed creating a native application but found that not to be feasible with the schedule we have. We started creating the design by drawing some mockups with different concepts. Ideally, we could have tested the various design ideas in a user test to see if the users thought the design were easy and intuitive to use. However, we ended up with not carrying it out because we wanted to combine our system testing at the same time as getting feedback for our web application. The final mockup designs can be seen in figures 3.9, 3.10, 3.11, and in Appendix B.1.2.

Map

For our service to provide a better understanding of where an entity resides, we have opted to incorporate an interactive map. The interactive map will have the ability to show pins and highlight the museums, restaurants other recommendations to the user. The map position will also serve as the location of the area to gather results. Upon entering the page, the map will be centered to show where the user is located given geolocation is available. From there and out, the user is free to discover other areas and their hidden treasures by changing the location of the map. To implement the interactive map into our web application, we looked many different frameworks which offer drawing functionality directly onto the map. Among these, we took a closer look at Google Maps and Leaflet.js.

Google Maps JavaScript API is one of the best enterprise options for displaying a map. Google also provides a well-documented and feature-rich library with tons of features[5]. Google also offers their API to the most popular platforms in use today. Unfortunately, one of the downsides of using Google Maps is that the usage of their API has its limitations and can be quite costly. The concern is not due to the potential initial usage of the service, but it might get expensive if the service gets popular.

Leaflet.js, on the other hand, is a lightweight open-source JavaScript framework for interactive maps[7]. It is designed simplicity and usability in mind and supports a vast range of map tile providers, such as Google Maps, OpenStreetMap[13], and Mapbox. After a close examination, it seemed easier to render custom icons and add features in forms of plugins on top of the Leaflet.js framework compared to the likes of Google Maps JavaScript API. The Contributors behind LeafletJS also speaks highly of it being mobile-friendly. Therefore, we have decided to go ahead with the usage of Leaflet.js as our services interactive map framework. Further, we opted to use Mapbox as our tile provider as we could customize our map layout and remove cluttered terrain that you had with OpenStreetMap.

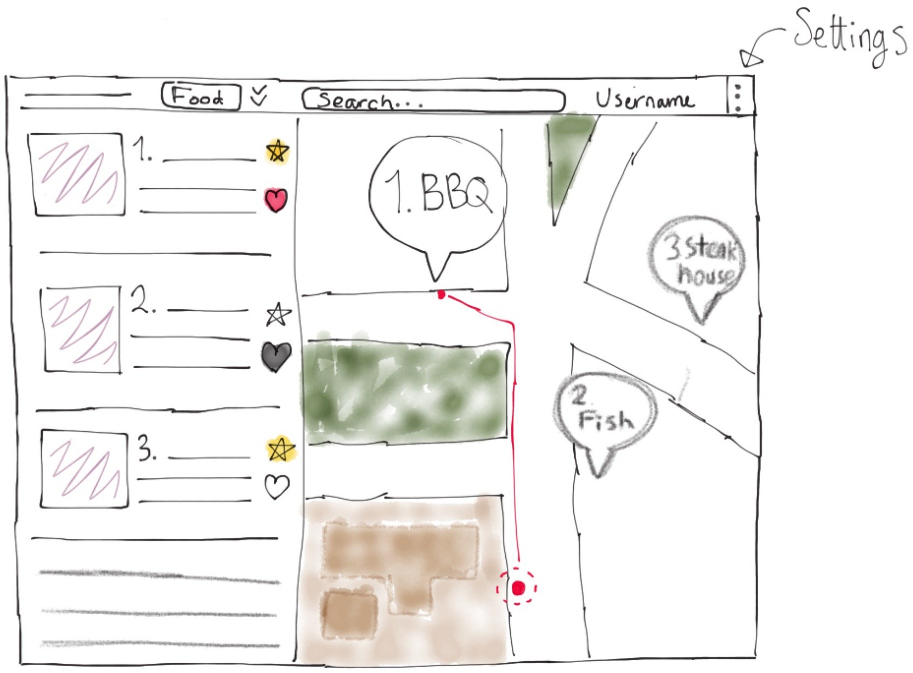


Figure 3.10: Proposed user interface with a map and a recommendation list

User Profile

The design for the user profile registration in the system is quite simple. On the registration page, the user will only need to fill in information about age, gender, and country and state of residence. See figure 3.9. From the mockup, we have removed the possibility for the user to register their email reason being we want as many users to register in the beginning, not make them feel like they are being monitored, and leave the usage anonymous. Additionally, to comply with local privacy laws. The user will also have the possibility to update their profile after the profile is created.

`User preferences`

The user has to be able to update their profile for the system to function properly from the beginning. As you can see from the figure 3.11, we have decided to use sliders for the user to manipulate their preferences. The users' preferences give the system an idea about a person interest and use it to provide recommendations. There is a small number of users using the service. The preferences are given on a rating between like(1), dislike(-1) or neutral(0).

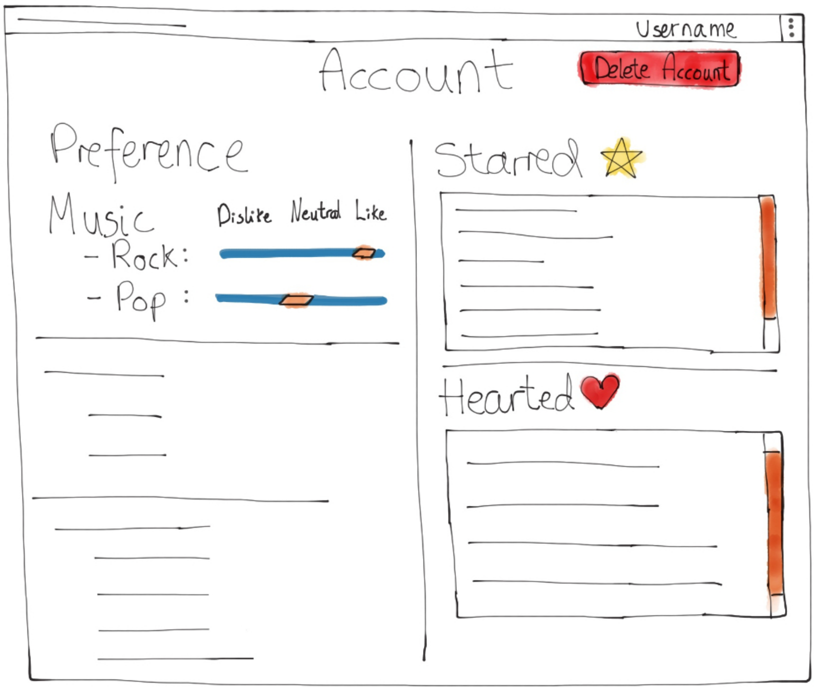


Figure 3.11: Proposed interface of the users profile page

`No Way To Trace The Users Back To One Individual`

We have designed the user profile in such a way that we do not store any information about the user which can identify the individual behind the account. Accordingly, we have chosen not to take their date of birth, marital status, email, and the users were encouraged under the user test to chose an ambiguous username. We also felt we had a higher chance of people participating and remain true to themselves if they were anonymous. In other words, their actions would not be compromised by privacy concerns. However, we do take approximate age groups as that helps with demographic filtering.

`Visited And Liked Places`
The user should also be able to see what activities and attractions they have visited and whom they would like to visit. It should also be possible to remove attractions and activities the users have visited and like from their list. From the mockup figure 3.11, we see how we envision the user to be able to see what attractions and activities they have visited and would like to visit.

3.4.2 Feedback

To help the system learn more about the user and to better understand their preferences, the system will require a method to get feedback from the users. After some consideration, we found that the best way to gather feedback from the user was through implicit feedback. We gather the implicit feedback from the attractions visited and liked from the users. The user will mark an attraction as visited by giving it a star, and the attraction as liked by giving it a heart.

Chapter 4 Implementation

In the following chapter, we will show in detail key technology used, our thought process for using said technology and the contributions it gives to the project. We will provide an overview of our high-level architecture and information flow. Further, we will discuss the main activities throughout our development process and some of the challenges we ran into as we progressed our work and ultimately what were our solutions to them.

4.1 Recommendation system

The recommendation system is made up of the following system components; a web server is running on Node.js, two databases powered by MongoDB and Redis respectively in addition to a Python module running a clustering algorithm. The overall system architecture can be viewed in figure 4.1.

4.1.1 Node.js

Our backend is running on Node.js[12], which is an event-driven, non-blocking I/O model JavaScript runtime built on V8 that makes it lightweight and efficient. V8 is part Google's open source project "The Chromium Project" and delivers a high-performance JavaScript engine. It is written in C++ and works by compiling JavaScript to native machine code before executing it. The compiled code is then dynamically optimized at runtime making Node.js an already thriving web server. Following Node.js' ecosystem is Node Package Manager (npm), a package manager making it easier to add dependencies to Node.js libraries. Today, most of the well recognized JavaScript libraries are available on npm in some way, shape or form. The package manager makes it easier to deploy code elsewhere.

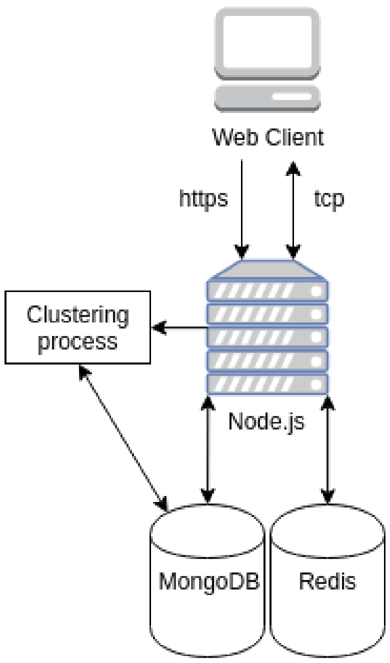


Figure 4.1: System Architecture

4.1.2 Express.js

Express.js[3] is our fast and minimalistic web framework for Node.js. It provides a thin layer of fundamental web application features. Features such as tools to handle HTTP request and responses, and routing options to create everything from simple web pages to RESTful services. Express.js offers functionality to add middleware to Express.js' runtime. Express.js is easily extendable with more features available as plugins, thus making Express.js our number one choice when building a modern web server.

Pug formerly known as Jade

Pug[15] is a high-performance template engine. The simple syntax allows writing web pages with ease. At runtime, the page will be rendered and served with Express.js in plain Hyper Text Markup Language (HTML). Another thing that makes pug exceptionally useful, and essentially why the page has to be compiled and served at runtime, is the ability for one template to inherit another template. This feature brings consistency across the board as your navigation menu markup, or any other typical boilerplate for that matter, can essentially be a standalone template. When the time comes to update it, you will not have to update manually every single page, but only the boilerplate code. Listing 4.1 shows pug's syntax and inheritance. Notice

```
extends layout

block content
    center
        h2 About

    div.col-xs-12.col-sm-12.col-md-8.col-md-offset-2
        center
            h3 Powered by:

        div.col-xs-12.col-sm-6.col-md-6
            center
                img(src="images/foursquare_logo.png")

        div.col-xs-12.col-sm-6.col-md-6
            center
                img(src="images/songkick_logo.png")
```

Listing 4.1: Pug template

how every aspect of standard HTML and boilerplate code is hidden away in "layout" and that the rest of the template is devoted to changing the main content for this specific page.

Passport

Passport.js[14] is our middleware authentication module for Express.js. In addition to supporting traditional username and password, it offers support with a set of authentication strategies such as Facebook, Twitter, OAuth, OpenID. We decided to keep everything local strategy and stick with username and salted and hashed password. We felt it gave us more freedom in the development process to write registration and login modules ourselves. Passport.js works seamlessly with MongoDB our database of choice. As with everything security related, you should avoid writing it yourself and Passport.js truly became a plug-and-play library. With the broad range of authentication strategies, we could easily expand to support popular third-party authentication services such as Facebook in the future should we want to.

4.1.3 MongoDB

MongoDB[9] is a NoSQL document-oriented database, which uses a JSON-like document structure. One of MonogDB's strengths is that not every document has to have the same structure. However, even if MongoDB does not enforce a schema, it is up to the application developer to structure and index the documents as it has a significant influence on the performance. MongoDB can easily be set up in a shared cluster. These two properties make MonogDB especially attractive for developers. In a matter of minutes, you have a feature rich database up and running locally. We decided to use MongoDB as our database being developer friendly and working seamlessly with Javascript.

Mongoose

We use mongoose.js[10] as our javascript bindings to MongoDB database. Mongoose.js provides with ways to extract, modify and store documents in our database. The API allows us to provide callbacks on most database operations, which not only makes operations themselves non-blocking to the Node.js runtime, but it also gives us flexible error handling should something occur.

4.1.4 Redis

In the project, we have opted to utilize Redis[16], which is an open source NoSQL in-memory data structure store. Redis supports a broad range of data structures and queries. Redis has built-in support among other things replication, Lua scripting, automatic cluster partitioning. Our aspiration was to optimize session and cookies handling with Node.js and settled with using Redis for its performance benefits in this regard. With the use of redis-connect as middleware in express.js and hosting our Redis server, the web application seems to run slightly more optimally than it otherwise would.

4.1.5 Socket.IO

Socket.IO is a real-time JavaScript library for web applications. It enables bidirectional event-based communication over the WebSocket protocol, which is a Transmission Control Protocol (TCP) connection in your browser. Socket.IO is split into two chunks: a client-side library and a server-side library for Node.js. They are nearly identical regarding API.

We opted to add and use Socket.IO for our web application. One of the challenges we faced mid-development was that we required getting our data from the server to the client. We ended up defining a few events the backend is listening on. The events make up a fixed data attributes attached which the backend expects with the broadcasted event. The events enable us to send and to receive data from the web client in the format we have specified ourselves.

4.1.6 Clustering

Our cluster algorithm is written in Python using scikit-learn and pymongo. We are primarily clustering our users using k-means clustering into a predefined number of clusters. The number of clusters is defined by equation 3.2. The primary reasons for using k-means clustering is that it's fast and give relatively good results. With pymongo, we are connecting to the MongoDB database to gather data from the user. These data of user preferences often referred to as features, are then translated into vector space. Scikit-learn uses these features and runs the k-means clustering. The results from scikit-learn are then written to our database. The recommendation algorithm works with these results, albeit how the clustering is implemented, by directly reading the cluster values from the database.

Scikit-learn

Scikit-learn is a straightforward and efficient tool for data mining and data analysis. It is built using NumPy, SciPy, and matplotlib. The library provides a broad range of implementations of algorithms to classifications, regressions, clusterings, dimensionality reduction, model selection and preprocessing. It has provided a huge benefactor to our timeline throughout the project.

4.1.7 Recommendation

The recommendation algorithm in place is rather simplistic. When a user queries an external data source, for a concerts, restaurants or other entities, the recommendation system returns the results of what the user is expecting along with any recommendation it has for the user. The proposed entities will show up in green text above the rest of the results. The current implementation does not distinguish stars and hearts to entities. The idea was valuing starts more than hearts, as stars meant that you had visited before and vouched for the place as opposed to heart said you were interested. See figure 4.3 for reference. Furthermore, we do not consider demographic variables such as age, gender or nationality nor the current location. You are very much able to search for entities all over the world, but the recommendations you receive are limited to those with you in the same cluster, and in those areas they have explored. Figure 4.2 explains how recommendations currently works. The solution we initially had foreseen was considering age as a demographic filter and having country distinct clusters and not the world as a whole. Moreover, we would only explore similar users locally as a country itself can be quite large. Lastly, the recommendations would be generated by mentioned pool of users of entities they have heart and star, and only the entities close to your specified location were considered. Every time you changed your profile or location, your cluster would automatically change to reflect your new profile.

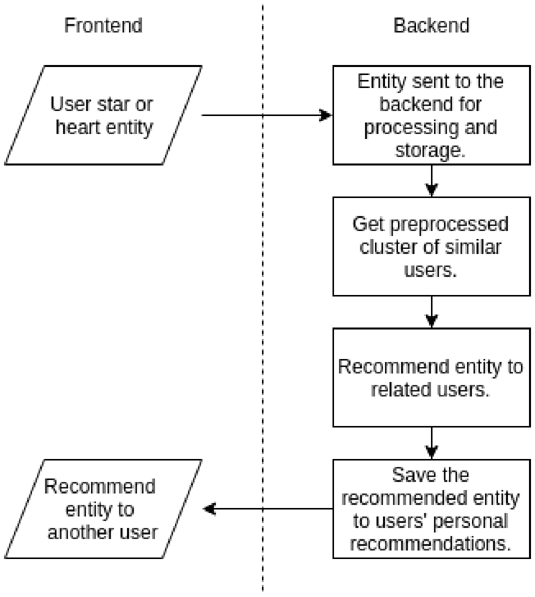

Frontend
Backend
User star or heart entity
Entity sent to the backend for processing and storage.
Get preprocessed cluster of similar users.
Recommend entity to related users.
Save the recommended entity to users' personal recommendations.
Recommend entity to another user

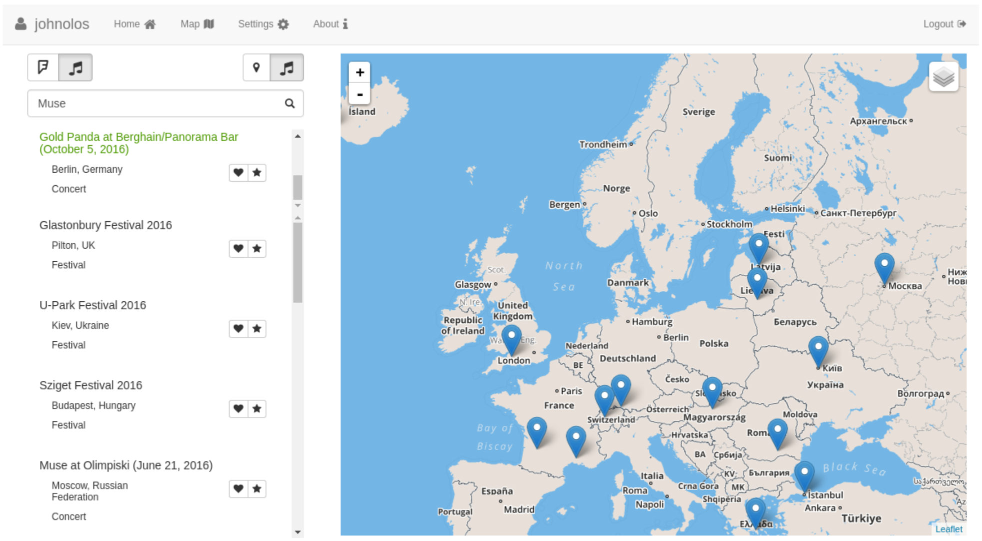
Figure 4.2: Flowchart for recommendations

4.2 Web application

Our client is made up of the following components.

4.2.1 Socket.io

As previously mentioned, we use Socket.IO both on the backend and the frontend. For the frontend, we have created a way for the user to query foursquare.com and songkick.com entities. The frontend then generates an event for query the particular service with data such as search phrase. The backend then receives the event with the attached data, performs whatever operations it has to do and returns the results back to the frontend. We have additionally specified a way for the user to alter the query depending on what the user is seeking. Songkick.com for example, we have support for both searching concerts and music related events locally without a search phrase or global search for a distinct artist. For foursquare.com, we only support local search with a given search phrase. See Appendix C.1 for a complete overview of the UI for the web application.



Gold Panda at Berghain/Panorama Bar (October 5, 2016)
Berlin, Germany
Concert

Glastonbury Festival 2016
Pilton, UK
Festival

U-Park Festival 2016
Kiev, Ukraine
Festival

Sziget Festival 2016
Budapest, Hungary
Festival

Muse at Olimpiski (June 21, 2016)
Moscow, Russian Federation
Concert

Figure 4.3: Screenshot of recommendation page

4.2.2 Angular[1]

AngularJS[2] extends HTMLs vocabulary and turns your static HTML documents with your angular code into a dynamic and expressive web application. The most prominent feature of angular is the two-way data binding. Angular.js detect changes in model section and triggers Document Object Model (DOM) manipulation when the values differ. Thus, your web page stays up-to-date without having to re-render the page. In-browser update works out neatly with our need for aggregating results from our backend to be displayed at out frontend. These results are usually not ready for the user at any given moment, and make the user wait or force a re-render later was simply not an option. We solved automatic re-rendering in Angular.js by writing a result service. An Angular.js service is a lazily instantiated and singleton application component that can be code shared across the application. You can see our simplified code achieve dynamic re-rendering in listing 4.2.

Last to be mentioned is an angular paradigm called directives. Directives are additional functionality which is not loaded by default. These directives can be standard angular libraries, or they can be third-party directives to make use easily of external libraries from within the application code. In the current project, we have used directives for Socket.IO and Leaflet.js, as they are both incredibly intertwined when it comes to information flow.

[1]We explicitly talk about Angular with respect to version 1 as version 2 is vastly different.

```
// Service in angular.
profileApp.factory('StarredStorage', function() {
    var starred = [];
    return {
        add: function(data) { starred.push(data); },
        get: function() { return starred; },
        remove: function(id) { starred.splice(id, 1); },
        set: function(data) { angular.copy(data, starred); }
    };
});

// Specifying a scope function to get data from service.
profileApp.controller('ResultsController',
        function($scope, StarredStorage, ... ) {
    $scope.starred = StarredStorage.get;
    ...
}

// Pug template getting data directly from service.
li(class="media" ng-repeat="star in starred()")
    div(class="media-body")
        h6 {{star.name}}
        div.col-xs-6.col-md-6
          ...
```

Listing 4.2: Dynamic re-rendering in Angular.js

4.2.3 Leaflet.js and Mapbox

Leaflet[7] is a mobile-friendly interactive map JavaScript library with an incredible community backing. Leaflet is well-documented and has a broad range of plugins as well as is easy to customize the way you want. The library itself is just a way to display, manipulate and add controls to the map, and you embed whatever tile service you want. For example, you can use Google Maps should you wish a familiar look for your users.

Mapbox

We opted to use our custom made tiles from a tile service called Mapbox[8]. Mapbox enabled us to tailor our tiles for our application by highlighting streets and paths, and we chose to remove cluttered information about terrain.

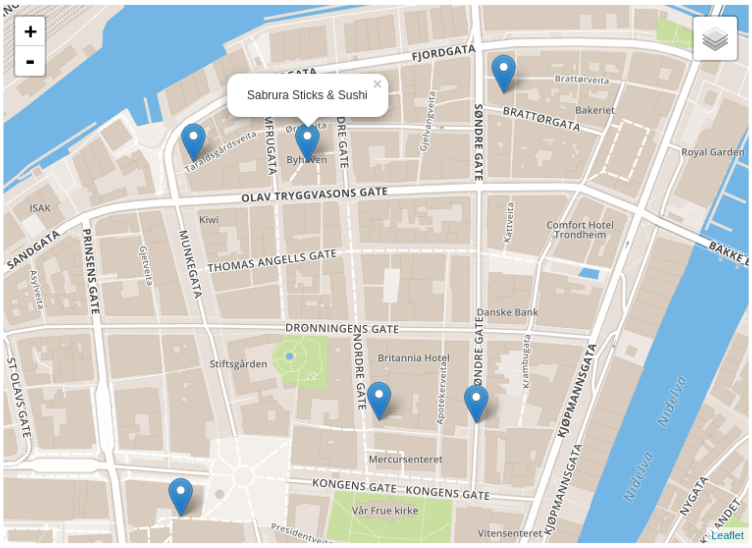


Figure 4.4: Leaflet map with Mapbox tiles

Leaflet in Angular

The folks at Google has recognized how popular Leaflet has become and maintains a Leaflet.js directive, ui-leaflet[1], which enables us to interact with the map through our angular application. The directive made it easy for us to center our map to an entity location when a user clicked on a result. Further, it made it easy for us to make pins on the map so that the user could clearly see the location of every result given it was known.

4.3 Challenges

These are some of the challenges we faced throughout our implementation. The challenges themselves are not particular advanced nor unique. However, we firmly believe that believe the path to creating a good web recommendation system involves solutions to these problems. There are countless solutions to these problems, and knowing how rapid JavaScript community evolves, there are up and coming libraries that solve these problems better than how we have archived. It remains to be said as it plays a big part in how our web application turned out.

4.3.1 In-Browser Update

Mid-development we recognized the complexity routing traffic between the server and the client turned out to be. At present, we did not consider Angular.js and Socket.IO as additions to our development stack.

We faced the issue that we wanted to run our cluster operations in the background, but at the time, we were forced to deal with them between the Hypertext Transfer Protocol (HTTP) requests and responses due to the limitations of the backend at the time. That yielded considerable extra response time to page requests were we wanted to make recommendations, which gave bad user experience. The solution became the use of Socket.IO. Socket.IO provided us with a way to send data to the client without having to re-render the page in addition to allowing us to send the data at any time.

Our second problem became how we could get our recommendation data from our socket at the client to render in the DOM. An alternative javascript library we considered early was React.js. React.js abstracts the DOM with a virtual one. That gives React.js the ability only to render the parts of the web page that have changed. However, due to the necessary modifications and completely re-writing our Pug-templates into React components, we decided to find another solution to our problem.

We were recommended using Angular.js as it gave us the ability to manipulate DOM on a high level. Not only was Angular.js a slightly less complicated library than React.js, but Angular.js also turned out to be compatible with our existing pug templates. Moving on, we were confident that our decision at the time was the right one.

Knowing what we know now in hindsight when choosing pug as our templating language, we might have spent the upfront time required to learn React.js, and more likely had a better time developing the web application in the later parts of the project. Needless to say, we're both happy with how it turned out.

4.3.2 Normalization of External Data Sources

It quickly became apparent when selecting our data sources that we would have to normalize to a unified format. Foursquare API and Songkick API had entirely different ideas of how to format their data even if their data to a large extent are similar. This is to be expected and didn't come off as a surprise.

Interesting key factors in our design decisions is to preserve the source id of the entity for the various external services. The mindset behind this decision is us having the option to link to the external services for more information if that was the direction we wanted to take the project. Another thing to note is that we kept categories to

```
{
    "name": "Name Of Entity",
    "source": "example.com",
    "source_id": "4575fc96bc9fec90c1a9a3eb3f560630",
    "location": {
        "lat": 63.4305,
        "lng": 10.3951,
        "city": "Trondheim, Norway"
    },
    "category": "Some Category",
}
```

Listing 4.3: Normalized data format

be a single value. On a general rule of thumb is that an entity has one category. A bank is a bank, and a hairdresser is and will always be a hairdresser. Complications arise when judging restaurants. A restaurant can be a “Hawaiian Barbeque” or a mixture of “Steakhouse” and “Sushi”. In such case, it would much rather benefit the recommendation system to have categories in an array of categories. We kept with one category to keep logic simple as such design decision affects both the backend and frontend. It remains to be said that for an ideal solution, every category is recorded into an array, and the categories themselves are normalized should synonyms exists. We will continue our discussion of normalizing categories in the future work section in the conclusion chapter. Listing 4.4 shows how we normalized data from foursquare.com.

```
function parse(raw_data) {
    var data = [];
    if(raw_data.hasOwnProperty('venues')) {
        for(var i = 0; i < raw_data.venues.length; i++) {
            var venue = raw_data.venues[i];
            var obj = {};
            obj.name = venue.hasOwnProperty("name") ?
                venue.name : "undefined";
            obj.source = 'foursquare.com';
            obj.source_id = venue.hasOwnProperty("id") ?
                venue.id : "undefined";
            if(venue.hasOwnProperty("location")) {
                obj.location = {};
                obj.location.lat =
                    venue.location.hasOwnProperty("lat") ?
                        venue.location.lat : "undefined";
                obj.location.lng =
                    venue.location.hasOwnProperty("lng") ?
                        venue.location.lng : "undefined";
                if(venue.location.hasOwnProperty("city") &&
                    venue.location.hasOwnProperty("country")) {
                    obj.location.city =
                        venue.location.city.concat(
                            ", ", venue.location.country
                        );
                }
            }
            if(venue.hasOwnProperty("categories")) {
                obj.category =
                    venue.categories.length > 0 ?
                        venue.categories[0].shortName : "undefined";
            }
            data.push(obj);
        }
    }
    return data;
}
```

Listing 4.4: Normalizing data from foursquare.com

Chapter 5

Evaluation

In this chapter, we will present our assessment of the recommendation system. We held a user experiment to see how the system operated under slight pressure, how easy the web application was to use and whether or not if the recommendations made was credible to the participants.

5.1 Current state

The implementation as it stands is a proof-of-concept as there were features left out which you would expect from such a system. A few of the features also had to be tweaked for the test environment, as sophisticated recommendation methods reviewed in 2 chapter is more feasible with a large number of users. Firstly, we could not get the system uptime required to evaluate it properly as such systems should be up and running for months before the gathered data is of high value, and such that the feedback would reflect the system credibility and performance. Secondly, even if we keep the system alive for such extended period, we wouldn't have any guarantee that the system would gather the attention from the large pool of users needed to get data which is credible. Finally, even if we did get a considerable number of users as we desire, we could not expect them to keep a high-level of dedication given the rough shape of the system.

A few of the planned tasks during the development of the system took longer than initially planned and other features had to be cut from the final system. Specifically, we removed the possibility to give recommendations in the categories movie and nightlife. We could not get this particular implementation done in time for the user test, and it is better not to support these categories if the recommendations are of a low grade. Hence, we chose to finish and improve other features needed in the service before running an experiment of the system with independent users.

5.2 User Experiment and System Testing

From a developers standpoint, recommendations are hard to verify without personal interpretation, and that led to the aspiration of a user experiment to collect feedback on how well the system performed. When the core functionality in the recommendation system was ready, we sent out invitations for a user experiment. In the experiment, we asked the participants to follow a number of step by step tasks to get them to know the system and its functionality. These step by step tasks take the user through account creation, changing their profile settings and all the way to using the system and receive recommendations. The experiment was used to see how the users would use the service, how easy they found it to use and see how the recommendations presented to the user performed and if they were credible to the participant. You can see the tasks the user had to perform in Appendix D.1.

As the final task, the participants had to answer a survey with questions about what their overall feeling and thoughts of the performance of the service were. Secondly, they were asked whether they thought of any improvements which could have been made to the system to improve their overall user experience. Although the results collected from the experiment were quite positive, it contained feedback listing features which they felt could be improved upon to enhance the overall user experience and system as a whole.

5.2.1 Validity of Experiment

When conducting an experiment, it is important to ensure that the validity of the research is as high as possible. High validity means the result would be as accurate as if you were to test on the general public. The two types of validity you will need to consider when conducting an experiment is both internal and external validity.

Internal Validity

An experiment has good internal validity if the measurements obtained are indeed due to your manipulations of the independent variable, and not to any other factors [44]. Most common threats to good internal validity is:

- **Differences between experimental and control group:** Any differences in the two groups may subsequently measure might not be attributable to your manipulation of the experimental group.

- **History:** Events you have not noticed might interfere between your pre-test and post-test observations.

- **Maturation:** The performance might have increased regardless of any manipulation you have done, the first test might give them practice and can affect the results.
- **Instrumentation:** Faulty instruments used to measure the dependent variable will affect the result.
- **Experimental mortality:** subjects might drop out before the study is finished due to a number of different reasons.
- **Reactivity and experimenter effects:** People might change their behavior as a reaction to being tested. Participants often want to help the researcher, or to look good, and so try to respond what they hope is the ‘right’ data

External Validity

An experiment has good external validity if the results collected are not unique to a particular set of circumstances, but are generalizable as in the same results can be predicted for subsequent occasions and in other situations. The best way of demonstrating generalizability is to repeat the experiment many times in different situations [44]. These are some of the main threats to external validity in experiments:

- **Over-reliance on special types of participants:** The use of students as subjects in the experiments. Since students are often younger and better educated than the general population, with different personal values and motivations. An experiments results might be generalizable to students, but it might not be a representation of the general population. Experiments who use volunteers are found that may not be generalizable to a wider population since the volunteers have a certain characteristic that differentiates them from the general population.
- **Too few participants:** The experiment does not have enough participants. Therefore, it is impossible to show that a result is statistically significant.
- **Non-representative participants:** The need to make sure that the group of participants is typical of the population that you wish to make statements about.
- **Non-representative test cases:** The data on which an experiment is based are typical of the kind of data files used in real-life

Our Experiments Validity

When we conducted the proof of concept experiment for our system, we asked our friends and fellow students to help us by taking part in the experiment. We ended up with having a medium internal validity for our experiment. Here are the reasons we have for saying the internal validity of the experiment is at a medium level:

- Experimental mortality is medium-high as all participants finished the experiment and answered the questionnaire afterward.
- Those who took part in the experiment were known to us before such the reactivity and experimenter effects were low since there exists a possibility for them to 'help', and hence not be completely honest with their responses although they were told to be completely honest both before and during the experiment. On the other hand, it is worth noting that they were completely anonymous and, in fact, encouraged to be so.
- The maturation of the experiment is also affected to be low-medium as most of the participants have heard about and seen parts of the system during the development phase, and it can affect the experiment in a negative manner.

The external validity of our experiment is at a low-to-medium level:

- Since we currently only rely on particular types of participants mostly students studying computer science, the results collected during the experiment can be affected to some degree as it does not test the general population's thoughts, but rather a particular part of society.
- The results collected during the research can also be criticized for a low number of participants.
- The participants in the experiment are in the goal group as young people often are the first people to embrace new services and applications.

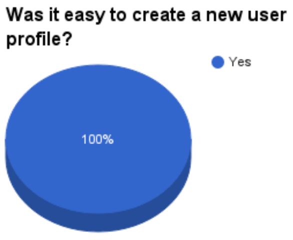
Even though the validity of our experiment is not at the highest level, we believe our experiment validity to be the best we could muster at the time it was conducted. Many of the people invited to participate declined due to being busy and did not have time to participate.

5.3 Experiment

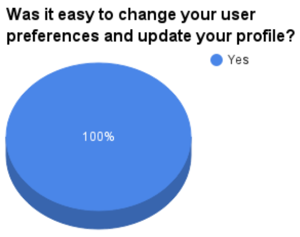
When the implementation of the recommendation system was finished, and all the functional requirements for the service to function were met, we set up an experiment to collect data from external users who were not involved in the development. This was done to test if the system we have implemented proved to function and give recommendations which appealed to the participants in the experiment. When the participants were done with the tasks, they were set to do they were asked to fill in a survey we had created using Google Survey. The survey was about their user experience, the results they received during the experiment and if they felt there were given credible recommendations. The experiment conducted had ten individuals participating in testing the service and leaving their thoughts on the system developed, as a proof-of-concept system.

You can see some of the results gathered from the experiment in Figure 5.1. As you can see, we got excellent feedback on both the creation and updating of the user profile to be easy and straightforward for the user. We worked especially hard with both designing and implementing the creation and updating of the user profile since we thought it might be a killer for the service if it were a daunting and a time-consuming task to complete for the user. The survey allowed the participants to leave written feedback about their experience with the system and the tasks they had to execute during the experiment, and it seems that they thought the User Interface (UI) looked good and was intuitive, but it could need some polish before being production ready to the general public. Among the written feedback, these messages were returning in most of the comments; "*Quite easy to use, but some of the symbols used was a bit hard to understand*", "*Yes it was easy to use. The system also provided suggestions I would normally not search for, which also is great.*", and "*Needs some polishing, but looks great!*".

The feedback we got about the UI is better than we could expect as we did not run a mockup trail on the UI at the time we finished creating and discussing the mockups. A potential mockup trial could have yielded better UI feedback in the experimental trial. Additionally, UI development is much more time costly than a mockup trial. However, we would have had to unique people participating in both of those as the experience of the UI could otherwise render the experiment feedback of the proof-of-concept with low validity.



(a) Answers from experiment about creating a new profile



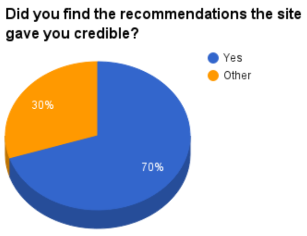
(b) Answers from experiment about updating user preferences

Figure 5.1: Answers from experiment about creation and updating user profile

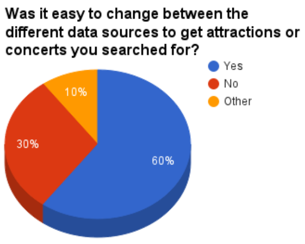
5.3.1 Experiment Recommendation Results

As you can see in figure 5.2a, most of the participants from the experiment found that the recommendations given to them to be credible, and they might want to visit the attraction or activity. Those of the participants who responded other in the survey left the feedback "*Sometimes, and sometimes not*" and "*Kind of*". Meaning that to most of the participants the recommendations were credible, but to some participants it was not. The feedback that some of the participants did not feel the recommendations provided not to be credible can be connected to the limited number of participants who were testing the system. Another reason can be that there is not enough data about the users and their history collected by the system at this time. If more data had been collected before the experiment such that the users could have been clustered more precise together, hence giving better recommendations to the users since there is more data to base the recommendations and clusters on. Therefore, the results from the user-user collaborative filtering sometimes can give unorthodox recommendations at the beginning of the system lifecycle.

Furthermore, an issue was that some of the test subjects answering that it was not intuitive enough how to change between the different data sources used in the system. This will need to be solved, see figure 5.2. This feedback might have been improved if we were to create a tutorial on how to use the system and what the different buttons in the system do and what their purpose is.



(a) Answers from experiment if recommendations was credible to user



(b) Answers from experiment about changing the source of the data

Figure 5.2: Answers from experiment about changing data source and credible recommendations

5.3.2 Evaluation of Clusters

The recommendation system relies on a clustering method to cluster users together into groups of people with similar interests and preferences. In our system we use k-means since it is fast and it provides good results, the downside of using k-means is the manually setting the number of clusters in the result. After testing k-means on the user data we collected using the survey we found that the equation 3.2 might produce the number of clusters needed for the service to function at a satisfactory level. One of the challenges we found when clustering user preferences and their history is to keep the cluster sizes at a satisfying level such that the recommendation process provides pleasing results toward the user. Therefore, the clusters need to be big enough for the system to provide good results at the same time as they should not be too big since the performance might drop when clusters are too big, and the provided user recommendations are too diverse.

There are other clustering methods we could have used in the system such as mini-batch k-means and BIRCH. Mini-batch k-means is faster than the original k-means method, and the resulting clusters are not that different from the ordinary k-means. However, the mini-batch k-means gives something back to the system performance wise if the number of samples is larger than 10k(10 000).[1] Our service would not get that many users for our experiment we opted not to use mini-batch k-means as the performance would not be that beneficial for the system and theoretically the results we would end up with is better with k-means. Another approach to consider is using the BIRCH clustering method. BIRCH provides good results when the dataset is large in addition to being a good method to use if the data requires a high number of clusters. It also finds the ideal number of clusters in runtime. Nevertheless, it is

[1]http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans

slower and requires more memory than k-means we felt that it was a bit overpowered at this stage. If the service was used by a large number of users, the change from k-means to BIRCH might be a good choice.

We have tried to visualize the user profiles from our experiment using k-means, mini-batch k-means, and BIRCH clustering, we used t-SNE to reduce the dimensions of the data. We have visualized the results both with running the clustering before the dimensionality reduction was applied, and the other way around. Trying to visualize and see if the resulting clusters used in the system was credible or not. We have reduced the dimensions into both two and three-dimensional spaces. We thought we could use the visualization to see if the clusters found were connected but discovered that with such low number of users it was hard to find groupings in the visualization. Some of the clusters also were scattered all over the space. However, we believe the reason for this to be the reduction of the data's dimensionality and the small number of users in the system. The visualizations can bee seen in Figure 5.3 and 5.4 and in Appendix D.2.

As you can see from the figures, k-means and mini-batch k-means gives few differences in where they place the different clusters. BIRCH finds more clusters in the data than the other two. This might mean that the participants preferences are diverse and might not fit together in a cluster. The results from the survey after the experiment implies that the participants found their recommendations credible. We believe the reason for this difference in data is due to the limited number of participants and that the vectors used to describe a user's preferences might not be scaled correctly when reflecting a user's preferences and history and that BIRCH supports a large number of clusters.

5.4 Evaluation and Discussion of the Implementation

One of our fears when conducting the experiment was that our developed service would not stay up or feel nonresponsive. We had not tested it with a larger user load before the experiment was conducted. Luckily, it performed great for all of the users, and they did not have any remarks during the experiment with the system crashing or not responding. The reason for this being one of our fears is that none of us have much experience with developing web applications and that we may have made some errors which we did not pick up during our testing of the service.

In hindsight, we probably could have chosen other frameworks which we have more experience with when creating the service. We did not have much experience with any of the more modern technologies in use today we opted to learn and use these for our service since they will perform much better and are much more scalable than the technologies we have used before. We used a lot of time getting to know how the technologies worked, but when we got the hang of it the production got very productive, and the functionality needed by the service were developed faster and faster. Even if the development got to a slow start, we are really happy for choosing these technologies and frameworks to work with, as we have learned a lot about how to develop a web-application with new and powerful frameworks.

5.5 Discussion of the Recommendation

The recommendation system we have developed is a user-user CF approach, a method which uses people with similar user profiles to give the user recommendations. The first step of the method is to group the users together into clusters of similar users. When a user requests recommendations, the method looks up people in the same cluster and their history to find recommendations for the user. This method might not be the best choice if the service was to be deployed on an industrial scale like TripAdvisor[2] or Stay.com,[3] as it can be some issues with scalability when the amount of data increases. To our knowledge, there has not been created a service like the one we are trying to prove is possible. Further, we felt that to give the users recommendations which were useful from the beginning we needed to base the recommendations provided using other users in your cluster have visited or liked.

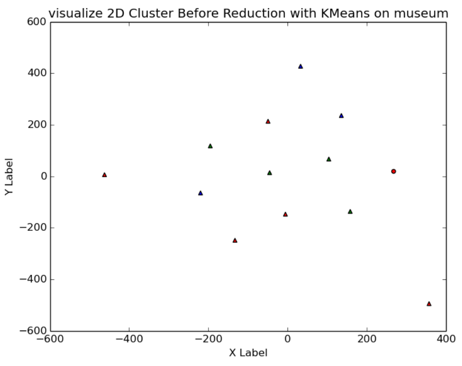
One of the main reasons for us not choosing the item-item CF was that we need to use APIs to get data about the different entities in the area of the user. This might cause the API usage being quite costly and also to beeing a bottleneck for the service as we do not store any data on our servers about the attractions or activities to comply with the agreements of the API. This especially comes into play when to iterate through entities and give recommendations to the users. If you were to change to item-item CF, we might need to reconsider creating our own data set within the different categories which we are fully in control to define and classify the categories.

Initially, we hoped to create a recommendation module with user-user CF and context-based approach, as we thought a hybrid approach would be the best way to give the best recommendations. As we had to use APIs as our data source, we have no way of influencing how the entities in the API is classified before we encounter them. Therefore, we did not find it plausible to finish creating the service with a hybrid recommendation system within our timeframe, and we would have needed a more dynamic approach to model the user in a vector space with the hybrid approach then we have currently. For these reasons, our primary choice of recommendation method is the user-user CF.

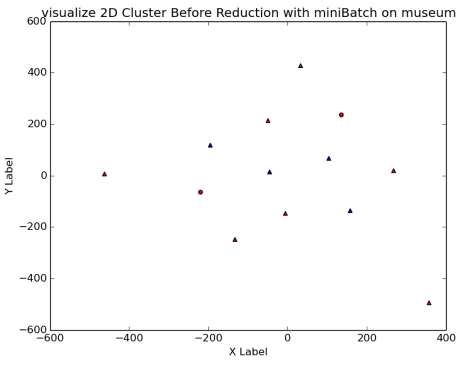
The recommendation module created need the particular domain it will give recommendations on for it to function, but the module itself is dynamic in such a way that the same module is used to provide recommendations in all of the categories.

[2]https://www.tripadvisor.com/
[3]http://www.stay.com/

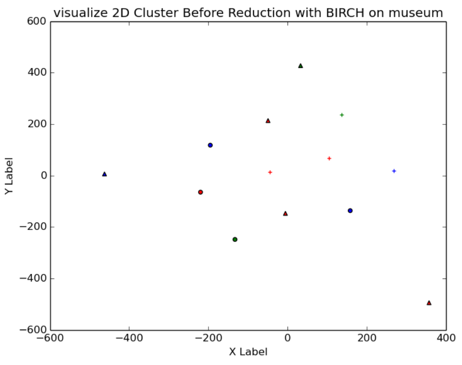
The feedback from our experiment shows that it works in most cases during the conducted experiment.



(a) K-Means

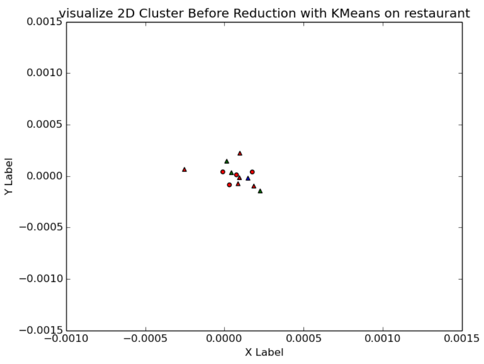


(b) Mini Batch K-Means

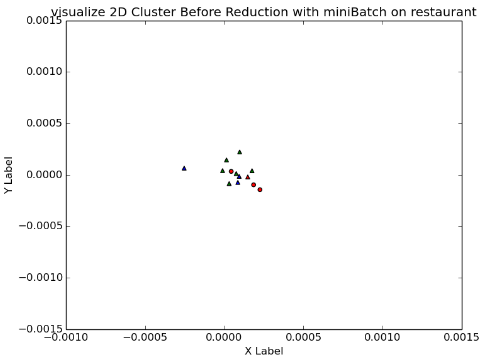


(c) BIRCH

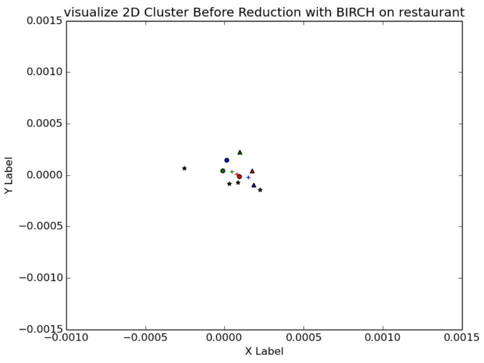
Figure 5.3: Visualization of clusters in category museum using K-Means, MiniBatch K-Means and BIRCH



(a) K-Means



(b) Mini Batch K-Means



(c) BIRCH

Figure 5.4: Visualization of clusters in category museum using K-Means, MiniBatch K-Means and BIRCH

Chapter 6 Conclusion

In the conclusion chapter, we will try to give the current status of our work at this time. Give a short summation of what we have done and what we originally envisioned for the project. We bring up the limitations in our service. Further, we list a few bullet points of what we believe is the way moving forward with what we have currently done. At the end of the chapter, we mention our final thoughts and conclusion.

6.1 Previous Work

There exists a lot of research in the area of recommendation systems and the different approaches used towards giving recommendations to the users of the system. To summarize, we have looked closer into these three domains:

- Content-Based Filtering
- Collaborative filtering
- Hybrid

There are many different areas in today's society which utilizes recommendations as a way give their user's suggestions they might not find on their own or would find interesting. Recommendation systems are used in the area of e-commerce, where it is used to give the user suggestions of what items they might want to buy. The recommendation methods are integrated with Netflix to help the users find movies they might like and to keep the user using their service. To our knowledge, there is no other service which uses one or more of these methods to provide users with recommendations on which attractions or activities they should visit in their vicinity.

Previously, Amazon[1] have used the same approach to give recommendations as we do in our system, but they have moved on to another type of CF in a combination with a

[1] https://www.amazon.com/

hybrid approach due to scalability and lack of robustness. A lot of the recommender systems currently being used are hybrid systems since most of the times a hybrid system will prove to be the best approach.

6.2 Visions for Proof-of-Concept System

When we started this research, we envisioned a proof-of-concept recommendation system for attractions and activities in the user's vicinity. To create this system we wanted to utilize a hybrid approach for our recommendation module. The hybrid approach we envisioned would use both context-based and user-user CF as we hoped this would provide the users with the best possible recommendations.

Further, we envisioned a recommendation module which would be dynamic in such a way that it would be capable of giving the users recommendation in all of the different domains. The module would require the user interest in a particular category for providing recommendations in that category for the user. The module will utilize the same methods for providing recommendations to the user across all domains. Hence, there is no specific module for recommendations in each of the categories.

Lastly, we envisioned to create a web application which could be used on any mobile or tablet platforms. The other platforms like mobile or tablets should contain all the methods and fields and not just a subset of the web application provided in a browser.

6.2.1 Limitations of the System

Although we had high expectations and visions in the beginning of the research for the system we wanted to check if viable. But there are some limitations to the system when we performed the experiment. However from the experimental results gathered we can see that the system performs well even with the limitations of the service. We will try to mention most of the limitations of the system in this section.

For the service to function as close to envisioned as possible with the limited time, we had to make a few compromises in the creation of the service. Consequently, we do not support the user to change their password and retrieve their username as these features is simply not implemented as they do not have any effect on how well the system can give personal recommendations.

As for user to receive recommendations, the user will currently have to wait until a user in the same cluster has either visited or liked an attraction or activity. These attractions will then be added to their recommendation list such that the user will keep these suggestions even if they were to change cluster at a later moment due to the system learning more about them by using the user's history. This is a limitation we made to the service, and it will not affect the results of our experiment with the time limit we conducted. However, for the service to be usable for tests over an extended period, changes are required to a more suitable method.

6.3 Future work

With the experiences made during this project, these bullet points are what we suggest for future work.

- Hybrid Approach: Improve the recommendation method used in the service to both utilize CF and Content-Based to provide better recommendations for the users. The Content-Based module might require the usage of natural language processing (NLP) to be able to categorize entities, and describe the metadata of the entities. This aligns with what we have seen in previous work and what state of the art recommendation systems are doing.
- Real-time recommendations: Fix the current recommendation module such that the recommendations are gathered in real-time when the user queries the service. Right now the recommendations performed ahead which became a compromise to a more easily experiment testing.
- Improved clustering: Implement a more dynamic way to model the users to achieve even better clustering. This entails looking at additional demographic attributes, cluster users geographically, and dynamic categories.
- User Interface: Improve the UI such that more users find it easy to use.
- Extensive experiments: Run an experiment over a longer period and a larger group of participants to maximize the experiment validity and check if results still are reasonable.

6.4 Conclusion and Final Thoughts

As a conclusion to our research with the results collected in our experiment in mind, we have concluded that we believe it to be possible to create a recommendation system which gives users suggestions into the different domains of attractions and activities. Although the research experiment we have conducted has few participants and a small timeframe we believe it shows that the recommendation system will work and can be used within these domains of attractions and activities. For us to give a better conclusion whether the recommendation system is viable or not, we believe that a longer and bigger experiment should be conducted to have more data to base our conclusion. However, with the current data from the experiments conducted so far we have an indication that the system is both feasible and yield reasonable good recommendations.

We believe that this type of recommender system could become a commercial success. Nowadays, typically young people could potentially be in the target group as they do not plan their trips ahead or at all. This kind of system can make it easier for them to plan while on the journey and help them find attractions or activities to visit in the vicinity when they have reached their destination. The current system developed values other people's opinions of what you should visit as long as you have similar preferences. Therefore, you would most likely be given a suggestion about a place you would like, but might not have visited unless you were recommended to go there. This service can become a valuable tool when traveling to new places looking for attractions and activities worthwhile to see and explore.

References

[1] Angular Leaflet Directive. https://github.com/angular-ui/ui-leaflet. Accessed: 2016-06-06.

[2] AngularJS - html enhancement for web apps. https://angularjs.org/. Accessed: 2016-06-06.

[3] Express.js is a minimal and flexible web framework for node.js. http://expressjs.com/en/index.html. Accessed: 2016-05-31.

[4] Foursquare developer Rate Limits. https://developer.foursquare.com/overview/ratelimits. Accessed: 2016-6-1.

[5] Google Map JavaScript API. https://developers.google.com/maps/documentation/javascript/. Accessed: 2015-12-15.

[6] Google movie showtime. https://www.google.com/movies. Accessed: 2016-6-1.

[7] LeafletJS is a lightweight javascript framework for interactive maps. http://leafletjs.com/. Accessed: 2015-12-15.

[8] Mapbox a mapping platform for developers. https://www.mapbox.com/. Accessed: 2016-05-30.

[9] MongoDB is a cross-platform document-oriented nosql database. https://www.mongodb.org/. Accessed: 2015-05-30.

[10] Mongoose.js elegant mongodb object modeling for node.js. http://mongoosejs.com/. Accessed: 2016-05-30.

[11] Nginx. https://nginx.org/en/. Accessed: 2016-6-8.

[12] Node.js is a javascript runtime built on chrome's v8 engine. https://nodejs.org/en/. Accessed: 2016-05-30.

[13] OpenStreetMap is a collaborative mapping service maintained by the community. https://www.openstreetmap.org/about. Accessed: 2015-12-15.

[14] Passport authentication for node.js. http://passportjs.org/. Accessed: 2016-05-31.

[15] Pug template engine. http://jade-lang.com/. Accessed: 2016-05-31.

[16] Redis.io. http://redis.io/. Accessed: 2016-06-01.

[17] Scikit-learn clustering simple and efficient tools for data mining and data analysis. http://scikit-learn.org/stable/modules/clustering.html. Accessed: 2016-5-2.

[18] Songkick. http://www.songkick.com/developer. Accessed: 2016-6-11.

[19] Supercharge your Node.js Applications with Nginx. http://blog.modulus.io/supercharge-your-nodejs-applications-with-nginx. Accessed: 2016-6-8.

[20] Tripadvisor API Access. https://developer-tripadvisor.com/content-api/request-api-access/. Accessed: 2016-6-1.

[21] Twitter Bootstrap is a html, css and js for developing responsive websites. http://getbootstrap.com/. Accessed: 2015-12-15.

[22] Yelp. https://www.yelp.com/developers/documentation/v2/overview. Accessed: 2016-6-11.

[23] J. Bobadilla, F. Ortega, a. Hernando, and a. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

[24] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and UserAdapted Interaction*, 12(4):331–370, 2002.

[25] Keunho Choi, Donghee Yoo, Gunwoo Kim, and Yongmoo Suh. A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis. *Electronic Commerce Research and Applications*, 11(4):309–317, 2012.

[26] Dorin Comaniciu and Peter Meer. Mean Shift: A Robust Approach toward Feature Space Analysis. *IEEE Transactions on patteren Analysis and machine intelligence*, 24(5):603–619, 2002.

[27] Konstantinos G Derpanis. Mean Shift Clustering. 1(2):1–4, 2005.

[28] Michael D. Ekstrand, John Riedl, and Joseph A. Konstan. Collaborative Filtering Recommender Systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173, 2010.

[29] Google. Google forms - create and analyze surveys, for free, https://www.google.com/forms/about/, 2015. [Online; accessed 15-December-2015].

[30] Yifan Hu, Chris Volinsky, and Yehuda Koren. Collaborative filtering for implicit feedback datasets. *Proceedings - IEEE International Conference on Data Mining, ICDM*, (July):263–272, 2008.

[31] Anil K Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[32] Kyoung-jae Kim and Hyunchul Ahn. A recommender system using GA K -means clustering in an online shopping market. 34:1200–1209, 2008.

[33] Mu Hsing Kuo, Liang C. Chen, and Chien W. Liang. Building and evaluating a location-based service recommendation system with a preference adjustment mechanism. *Expert Systems with Applications*, 36(2 PART 2):3543–3554, 2009.

[34] T Lee, Y Park, and Y Park. A time-based approach to effective recommender systems using implicit feedback. *Expert Systems with Applications*, 34(4):3055–3062, 2008.

[35] Greg Linden, Brent Smith, and Jeremy York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *{IEEE} Internet Computing*, 7(1):76–80, 2003.

[36] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[37] Laurens Van Der Maaten, Eric Postma, and Jaap van den Herik. Dimensionality Reduction: A Comparative Review. *J Mach Learn Res*, 10:66–71, 2009.

[38] Chavant Megana M, Patil Asawari, Davil Lata, and Patil Ajinkya. Mini Batch K-Means Clustering On Large Dataset. 04(07):1356–1358, 2015.

[39] Robin Van Meteren and Maarten Van Someren. Using Content-Based Filtering for Recommendation. *ECML/MLNET Workshop on Machine Learning and the New Information Age*, pages 47–56, 2000.

[40] Alexandros Nanopoulos, Milos Radovanović, and Mirjana Ivanović. On the Existence of Obstinate Results in Vector Space Models. *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, (Section 2):186—-193, 2010.

[41] Hai Thanh Nguyen, Thomas Almenningen, Martin Havig, Helge Langseth, and Heri Ramampiaro. Learning to Rank for Personalised Fashion Recommender Systems via Implicit Feedback. pages 51–61, 2014.

[42] Edward Rolando Núñez-Valdéz, Juan Manuel Cueva Lovelle, Oscar Sanjuán Martínez, Vicente García-Díaz, Patricia Ordoñez De Pablos, and Carlos Enrique Montenegro Marín. Implicit feedback techniques on recommender systems applied to electronic books. *Computers in Human Behavior*, 28(4):1186–1193, 2012.

[43] Douglas W Oard and Jinmook Kim. Implicit Feedback for Recommender Systems. *Proceedings of the AAAI workshop on recommender systems*, pages 81–83, 1998.

[44] Briony J Oates. *Researching information systems and computing.* Sage, 2005.

[45] Dan Pelleg and Andrew Moore. Accelerating exact k-means algorithms with geometric reasoning. pages 277–281, 1999.

[46] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. pages 727–734, 2000.

[47] Shaghayegh Sahebi and William W Cohen. Community-based recommendations: a solution to the cold start problem. *Workshop on Recommender Systems and the Social Web (RSWEB), held in conjunction with ACM RecSys'11,*, 2011.

[48] Oscar Sanjuan Martínez, Cristina Pelayo G-Bustelo, Ruben González Crespo, and Torres Franco Enrique. Using Recommendation System for E-learning Environments at degree level. *International Journal of Artificial Intelligence and InteractiveMultimedia.*, 1:67–70, 2009.

[49] Seth Sorensen. Accuracy of Similarity Measures in Recommender Systems. 2012.

[50] Michael Steinbach, G Karypis, and V Kumar. A Comparison of Document Clustering Techniques. *KDD workshop on text mining*, 400:1–2, 2000.

[51] Michael Steinbach, Vipin Kumar, and Levent Ertöz. The Challenges of Clustering High Dimensional Data *. *New Directions in Statistical Physics*, pages 273–309, 2004.

[52] Loren Terveen and Will Hill. Beyond recommender systems: Helping people help each other. *HCI in the New Millennium*, (1):1–21, 2001.

[53] Alexander Tuzhilin and Gediminas Adomavicius. Profiling in Personalizat ion Applications Rule Discovery and Validat ion. *New York*, pages 377–381.

[54] Pei Wang. Why recommendation is special. *Workshop on Recommender Systems, part of the 15th National Conference on Artificial Intelligence (AAAI 1998)*, pages 111–113, 1998.

[55] WEKA. Weka 3 - data mining with open source machine learning software in java, http://www.cs.waikato.ac.nz/ml/weka/, 2015. [Online; accessed 15-December-2015].

[56] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, June 1996.

Appendix A
Survey

A.1 Example of Calculating Rating Using Pearson correlation

	Batman Begins	Alice in Wonderland	Dumb and Dumber	Equilibrium
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?
User D	2	4	?	3
User E	3	4	5	?

Table A.1: A rating matrix on a 5-star scale [28]

A example from Ekstrand[28] of how to calculate User C predicted rating of the movie Equilibrium, we will use the data is from the table A.1:

- Pearson Correlation
- Neighborhood size of 2
- Weighted average with mean offset

Cs mean rating is $3,667$, and there is only two users who have rated Equilibrium, therefore the two users are used as the neighborhood $s(C,A) = 0,832$ and $s(C,D) = -0,515$.

$$\begin{aligned} P_{C,e} &= \bar{r}_C + \frac{s(C,A)(r_{A,e} - \bar{r}_A) + s(C,D)(r_{D,e} - \bar{r}_D)}{|s(C,A)| + |s(C,D)} \\ &= 3,667 + \frac{0,832 * (5-4) + -0,515(2-3)}{0,832 + 0,515} \\ &= 4,667 \end{aligned} \tag{A.1}$$

A.2 X-Means Clustering

X-means is an improved version of the k-means which utilizes statistical modeling to estimate the optimized number of clusters. The estimation processes find n, the number of clusters, at the end of each iteration in k-means. Local decisions are about which subset of the current centroids should split themselves to fit the data better [46]. The decision to divide the centroids is made by calculating the Bayesian Information Criterion (BIC), see equation A.2. $\hat{l_j}(D)$ is the log-likely-hood of the data according to the j-th model and taken the maximum likely-hood point, and p_j is the number of parameters in M_j

$$BIC(M_j) = \hat{l_j}(D) - \frac{p_j}{2} * \log R \tag{A.2}$$

The proposed X-means scales better than iterated k-means. The result from the experiments shows that X-means runs twice as fast on large problems [46].

Algorithm A.1 X-Means algorithm

```
Require: $X_i$ elements described by m attributes, i: number of centroids and I:
    Upper bound
1: Improve-Params: Run conventional K-Means
2: Improve-Structure: Find where new centroids should appear. Done by letting
   the centroids split in two, this is done using two strategies.
3: Strategy one One at a time: pick one centroid split it by placing a new centroid
   nearby, run K-means and see if model score is better. If better accept new centroid,
   if not return to previous structure
4: Strategy two Try half the centroids: Choose half the centroids using heuristic
   criterion on how good a split will be.
5: if i > $I_{max}$ then
       Stop and report the best scoring model found during the search
6: else
       Goto 1
7: end if
```

A.3 Visualization

A table containing different dimensionality reduction methods used to visualize high-dimensional data, it shows what parameters is needed to run the method, the runtime of the method and how much memory needed for each method.

Technique	Paramteric	Parameters	Computational	Memory
PCA	yes	none	$O(D^3)$	$O(D^2)$
Class. scaling	no	none	$O(n^3)$	$O(n^2)$
Isomap	no	k	$O(n^3)$	$O(n^2)$
Kernel PCA	no	k(.,.)	$O(n^3)$	$O(n^2)$
MVU	no	k	$O((nk)^3)$	$O((nk)^3)$
Diffusion maps	no	σ, t	$O(n^3)$	$O(n^2)$
LLE	no	k	$O(pn^2)$	$O(pn^2)$
Laplacian Eigenmaps	no	k, σ	$O(pn^2)$	$O(pn^2)$
Hessian LLE	no	k	$O(pn^2)$	$O(pn^2)$
LTSA	no	k	$O(pn^2)$	$O(pn^2)$
Sammon mapping	no	none	$O(in^2)$	$O(n^2)$
Autoencoders	yes	net size	$O(inw)$	$O(w)$
LLC	yes	m, k	$O(imd^3)$	$O(nmd)$
Manifold charing	yes	m	$O(imd^3)$	$O(nmd)$

Table A.2: Properties of techniques for dimensionality reduction [37]

A.3.1 Convex and Non-Convex Dimensionality Reduction

Convex

Convex techniques for dimensionality reduction optimizes the objective function used in the technique, the objective function does not contain a local optima i.e the space is convex[37]. The objective function used in these techniques usually has the form of a Rayleigh quotient: $\phi(Y) = \frac{Y^TAY}{Y^TBY}$. This objective function is a well known function and it can be optimized by solving it's generalized eigenproblem. The convex dimensionality reduction techniques can be divided into two different subcategories those who perform the eigendecomposition of a full matrix and those who only perform it of a sparse matrix.

Techniques performing optimization on full matrix:

1. PCA / Classical scaling
2. Isomap
3. Kernel PCA
4. Maximum Variance Unfolding
5. Diffusion maps

Techniques performing optimization on sparse matrix:

1. LLE
2. Laplacian Eigenmaps
3. Hessian LLE
4. LTSA

Non-Convex

Techniques that construct low-dimensional data by optmizing a convex objective function by means of an eigendecomposition.

Techniques performing optimization:

1. Sammon Mapping
2. Multilayer Autoencoders
3. LLC
4. Manifold Charting

A.3.2 MNIST Data Set Visualization

Visualization of MNIST Data set using different dimensionality reduction techniques.

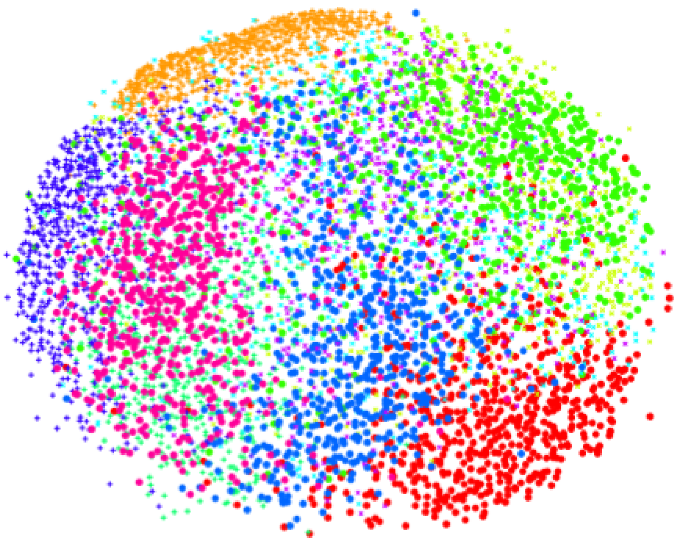


Figure A.1: Visualization by Sammon mapping of the MNIST data set [36]

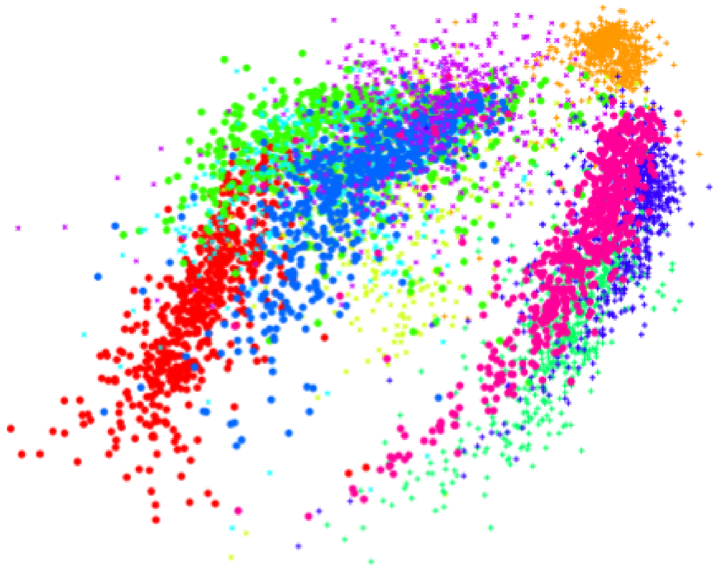


Figure A.2: Visualization by Isomap of the MNIST data set [36]

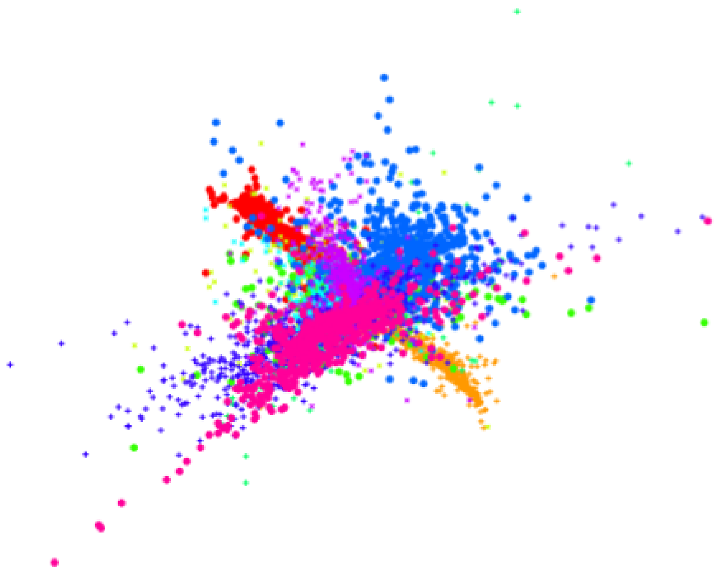


Figure A.3: Visualization by LLE of the MNIST data set [36]

Appendix B

Approach

B.1 Requirements

B.1.1 User Profile Requirements

The categories the user will have to give their preferences in to create a user profile in the system:

- Personal information
  - Age-group
    - 0-9
    - 10-19
    - 20-29
    - 30-39
    - 40-49
    - 50-59
    - 60-69
    - 70+
  - Gender
    - Man
    - Woman
    - Other
  - Country
  - State
- Movies
  - Action
  - Adventure
  - Animation
  - Biography
  - Comedy
  - Crime
  - Documentary
  - Drama
  - Family
  - Fantasy
  - History
  - Horror
  - Musical
  - Mystery
  - Romance
  - Sci-Fi
  - Sports
  - Thriller
  - War
  - Western
- Music

  - Alternative
  - Blues
  - Children music
  - Classics
  - Country
  - Dance & EDM
  - Electronic
  - Hip-hop and Rap
  - Jazz
  - Opera
  - Pop
  - R&B and Soul
  - Reggae
  - Rock
- Restaurants
  - Burger
  - Café
  - Scandinavian
  - General food
  - Italian
  - Sushi
  - BBQ
  - Indian
  - Mexican
  - Vegetarian
  - Steakhouse
  - Tapas
  - Chinese & Oriental
- Museums
  - Arts and Architecture museum
  - Children museum
  - Theme museum
  - History museum
  - Military museum
  - Science museum
- Nightlife
  - Pub
  - Brewery
  - Bar
  - Sports bar
  - Dive bar
  - Wine bar
  - Lounge
  - Nightclub

B.1.2 Interface

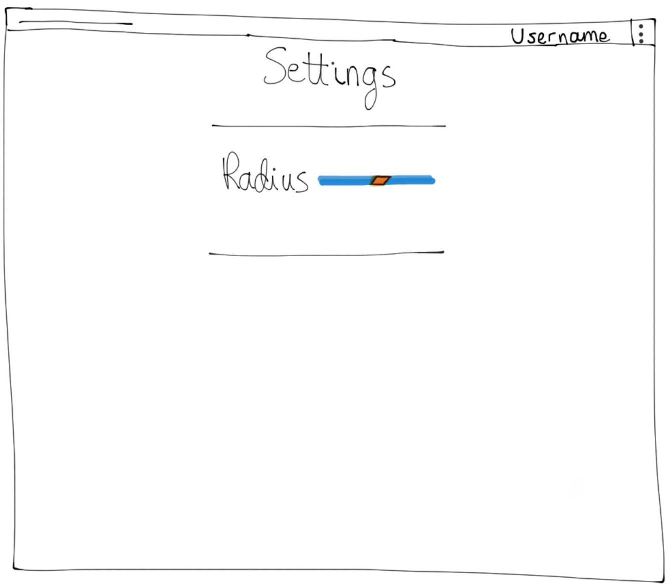


Figure B.1: Settings interface mockup

B.1.3 Sequence Diagrams

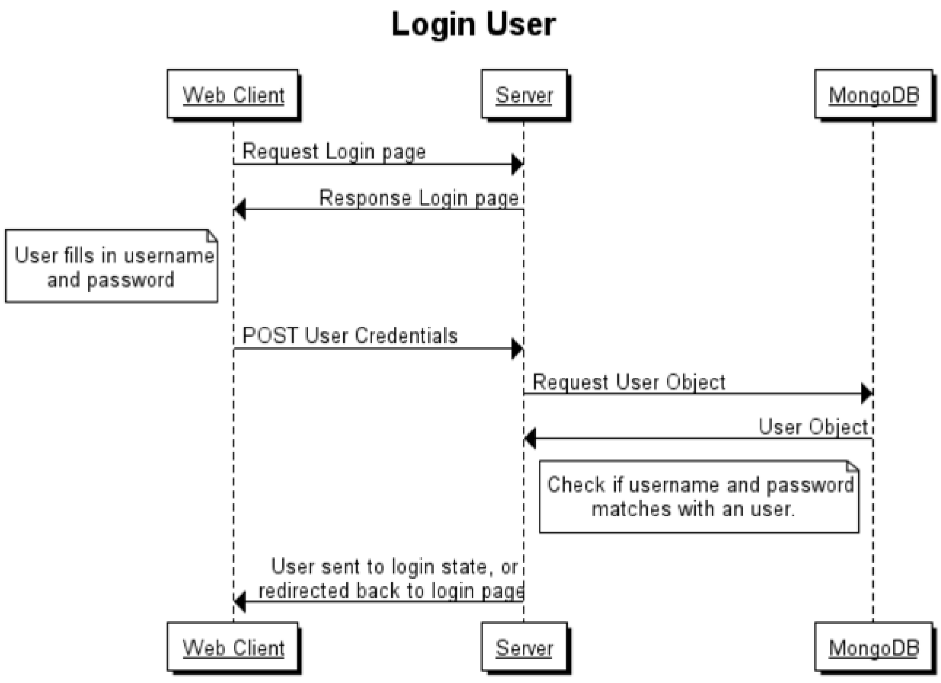

Login User
Web Client
Server
MongoDB
Request Login page
Response Login page
User fills in username and password
POST User Credentials
Request User Object
User Object
Check if username and password matches with an user.
User sent to login state, or redirected back to login page
Web Client
Server
MongoDB

Figure B.2: A sequence diagram for when users login into the system

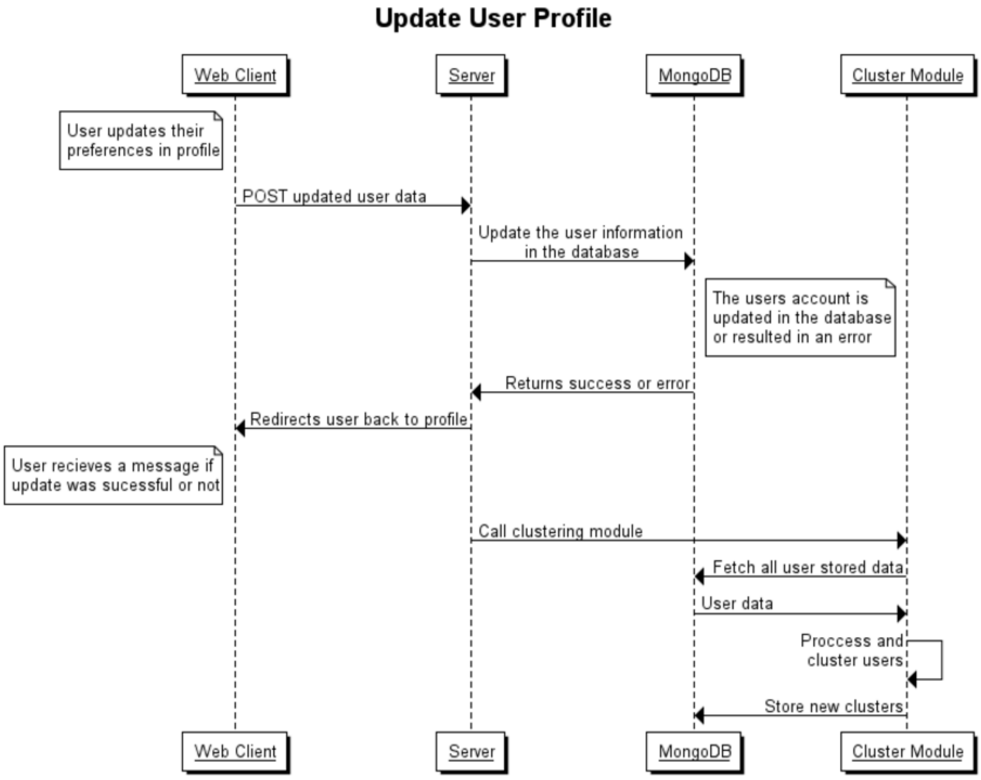

Update User Profile
Web Client
Server
MongoDB
Cluster Module
User updates their preferences in profile
POST updated user data
Update the user information in the database
The users account is updated in the database or resulted in an error
Returns success or error
Redirects user back to profile
User recieves a message if update was sucessful or not
Call clustering module
Fetch all user stored data
User data
Proccess and cluster users
Store new clusters
Web Client
Server
MongoDB
Cluster Module

Figure B.3: A sequence diagram for when users updates their profile

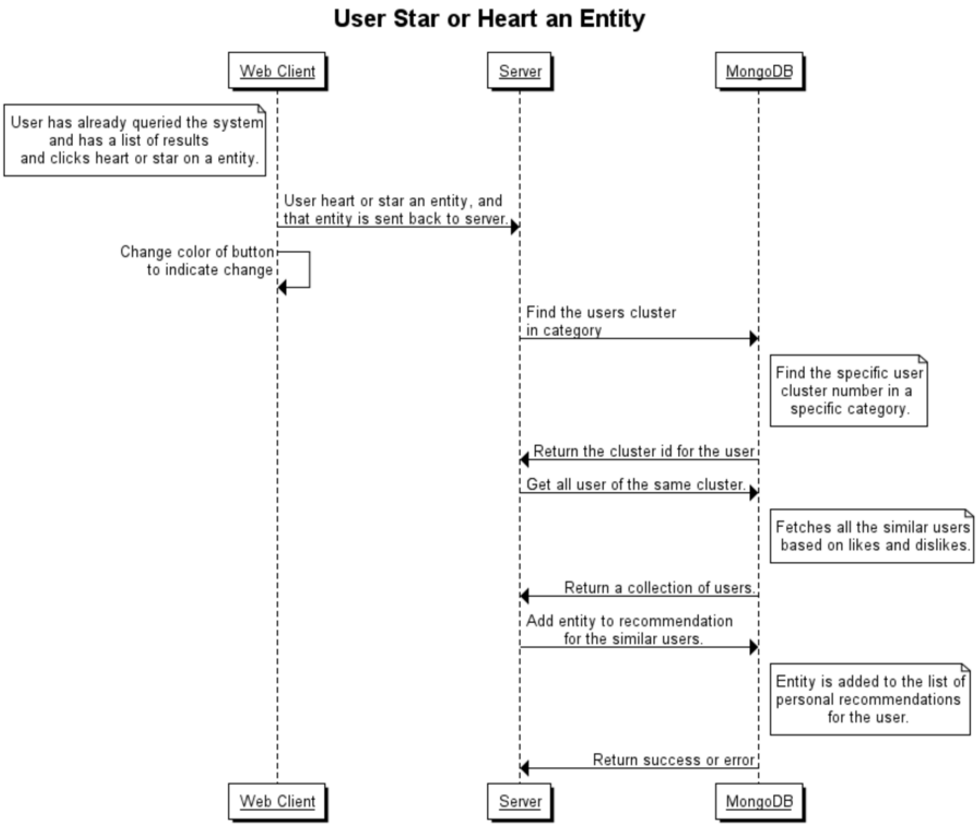

User Star or Heart an Entity
Web Client
Server
MongoDB
User has already queried the system and has a list of results and clicks heart or star on a entity.
User heart or star an entity, and that entity is sent back to server.
Change color of button to indicate change
Find the users cluster in category
Find the specific user cluster number in a specific category.
Return the cluster id for the user
Get all user of the same cluster.
Fetches all the similar users based on likes and dislikes.
Return a collection of users.
Add entity to recommendation for the similar users.
Entity is added to the list of personal recommendations for the user.
Return success or error
Web Client
Server
MongoDB

Figure B.4: A sequence diagram for when user hear or star an entity

B.1.4 Clustering

Clustering tests

Number of clusters	**Sum of squared errors**	**Iterations used**	**Time to build model[sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	799	4	0 sec	82	134
3	723	3	0.03 sec	57	94
4	646	3	0 sec	38	71
5	590	5	0 sec	16	69
6	589	5	0.01 sec	1	69
7	561	5	0 sec	1	59
8	547	5	0 sec	1	52
9	506	7	0 sec	13	45
10	525	6	0 sec	1	53
11	537	6	0 sec	1	60
12	516	6	0.01 sec	1	54
13	506	6	0 sec	1	54
14	479	6	0 sec	1	45
15	471	5	0 sec	1	45
20	440	5	0.01 sec	1	37
25	384	7	0.02 sec	1	23
30	357	6	0.01 sec	1	19
40	304	6	0.01 sec	1	16

Table B.1: Simple K-Means on Museums with sex, age and marital-status using Euclidean distance, max iterations: 500 and 10 seeds using WEKA

Number of clusters	**Sum of squared errors**	**Iterations used**	**Time to build model[sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	629	4	0 sec	83	133
3	520	4	0 sec	46	93
4	505	2	0 sec	20	88
5	465	4	0 sec	10	66
6	419	5	0 sec	10	65
7	400	5	0 sec	10	64
8	384	5	0 sec	10	58
9	366	5	0 sec	8	56
10	355	3	0 sec	8	55
11	346	3	0 sec	6	53
12	336	3	0 sec	6	51
13	320	3	0 sec	6	51
14	311	3	0 sec	6	51
15	302	3	0 sec	5	51
20	251	5	0 sec	4	29
25	236	4	0.01 sec	1	30
30	218	5	0.01 sec	1	27
40	174	3	0.02 sec	1	18

Table B.2: Simple K-Means on Museums without sex, age and marital-status using Euclidean distance, max iterations: 500 and 10 seeds using WEKA

Number of clusters	**Sum of squared errors**	**Iterations used**	**Time to build model[sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	1626	5	0 sec	82	134
3	1520	4	0 sec	61	93
4	1461	6	0 sec	41	82
5	1374	8	0 sec	26	64
6	1343	6	0 sec	20	52
7	1311	5	0 sec	22	46
8	1261	6	0 sec	20	46
9	1229	7	0.01 sec	14	45
10	1209	8	0.04 sec	10	42
11	1186	7	0.01 sec	8	41
12	1158	6	0.01 sec	6	35
13	1139	6	0.01 sec	6	28
14	1120	8	0.01 sec	6	24
15	1103	7	0.01 sec	7	24
20	1012	8	0.02 sec	6	21
25	1000	6	0.01 sec	1	22
30	911	7	0.03 sec	1	17
40	822	7	0.02 sec	1	14

Table B.3: Simple K-Means on Music with sex, age and marital-status using Euclidean distance, max iterations: 500 and 10 seeds using WEKA

Number of clusters	**Sum of squared errors**	**Iterations used**	**Time to build model[sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	1285	5	0 sec	79	137
3	1219	5	0 sec	60	87
4	1127	7	0 sec	41	72
5	1114	5	0 sec	19	61
6	1069	5	0 sec	24	65
7	1026	5	0 sec	19	48
8	997	5	0 sec	18	49
9	997	4	0 sec	10	45
10	970	7	0 sec	11	39
11	936	8	0.01 sec	12	36
12	872	10	0.02 sec	8	42
13	887	6	0.01 sec	9	25
14	841	6	0.01 sec	7	25
15	827	6	0.01 sec	8	23
20	771	5	0.01 sec	3	20
25	728	6	0.01 sec	2	21
30	687	6	0.01 sec	1	18
40	624	7	0.01 sec	1	14

Table B.4: Simple K-Means on Music without sex, age and marital-status using Euclidean distance, max iterations: 500 and 10 seeds using WEKA

Number of clusters	**Sum of squared errors**	**Iterations used**	**Time to build model[sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	959	4	0 sec	81	135
3	868	5	0 sec	44	97
4	846	4	0 sec	52	77
5	762	5	0 sec	20	62
6	742	5	0 sec	18	56
7	710	5	0 sec	18	54
8	674	6	0 sec	16	50
9	660	5	0 sec	11	44
10	628	4	0 sec	14	46
11	597	4	0 sec	12	39
12	563	5	0 sec	8	31
13	561	5	0 sec	2	32
14	555	4	0 sec	1	35
15	556	4	0.01 sec	1	34
20	509	4	0.01 sec	1	24
25	474	5	0.01 sec	1	19
30	428	5	0.01 sec	1	17
40	388	5	0.01 sec	1	14

Table B.5: Simple K-Means on Nightlife with sex, age and marital-status using Euclidean distance, max iterations: 500 and 10 seeds using WEKA

Number of clusters	**Sum of squared errors**	**Iterations used**	**Time to build model[sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	738	4	0 sec	51	165
3	526	6	0 sec	46	98
4	454	6	0 sec	18	90
5	420	7	0 sec	17	71
6	447	5	0 sec	11	62
7	414	5	0 sec	11	74
8	401	5	0 sec	9	74
9	387	5	0 sec	7	67
10	350	5	0 sec	6	64
11	348	4	0 sec	5	66
12	344	4	0 sec	5	65
13	340	4	0 sec	2	65
14	335	4	0 sec	2	65
15	325	4	0 sec	2	54
20	271	4	0 sec	1	33
25	231	5	0 sec	1	27
30	215	4	0.01 sec	1	26
40	187	5	0.01 sec	1	26

Table B.6: Simple K-Means on Nightlife without sex, age and marital-status using Euclidean distance, max iterations: 500 and 10 seeds using WEKA

Number of clusters	**Sum of squared errors**	**Iterations used**	**Time to build model[sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	1134	3	0 sec	59	157
3	1054	6	0 sec	51	106
4	968	4	0 sec	42	68
5	950	4	0 sec	11	62
6	922	9	0.01 sec	10	60
7	902	9	0.01 sec	9	58
8	917	9	0.01 sec	1	62
9	888	9	0.01 sec	1	59
10	875	5	0.01 sec	1	59
11	851	5	0 sec	1	59
12	813	10	0.01 sec	1	57
13	800	10	0.01 sec	1	51
14	794	4	0.01 sec	1	38
15	769	5	0.01 sec	1	32
20	728	5	0.01 sec	1	33
25	681	5	0.01 sec	1	28
30	636	6	0.01 sec	1	23
40	563	7	0.02 sec	1	18

Table B.7: Simple K-Means on Restaurant with sex, age and marital-status using Euclidean distance, max iterations: 500 and 10 seeds using WEKA

Number of clusters	**Sum of squared errors**	**Iterations used**	**Time to build model[sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	819	3	0 sec	57	159
3	749	3	0 sec	52	112
4	677	3	0 sec	44	65
5	653	6	0 sec	13	58
6	627	6	0.01 sec	12	61
7	626	6	0.01 sec	6	62
8	610	5	0 sec	6	62
9	600	5	0.01 sec	4	62
10	586	4	0 sec	4	62
11	572	4	0 sec	4	51
12	551	5	0 sec	4	60
13	539	5	0 sec	4	60
14	522	4	0.01 sec	4	44
15	523	4	0 sec	3	42
20	490	7	0.02 sec	1	50
25	448	7	0.01 sec	1	50
30	405	8	0.03 sec	1	26
40	362	7	0.01 sec	1	26

Table B.8: Simple K-Means on Restaurant without sex, age and marital-status using Euclidean distance, max iterations: 500 and 10 seeds using WEKA

Number of clusters	**sum of squared errors**	**Iterations used**	**Time to build model[sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	5513.0	10	0.1 seconds	106	110
3	5209.0	14	0.17 seconds	52	85
4	5074.0	11	0.24 seconds	48	71
5	4947.0	9	0.17 seconds	30	74
6	4878.0	11	0.32 seconds	16	55
7	4771.0	17	0.24 seconds	14	51
8	4741.0	9	0.19 seconds	8	51
9	4640.0	7	0.11 seconds	8	37
10	4553.0	7	0.16 seconds	6	38
11	4502.0	9	0.32 seconds	5	34
12	4455.0	7	0.15 seconds	7	35
13	4411.0	9	0.27 seconds	3	41
14	4361.0	10	0.31 seconds	3	35
15	4354.0	10	0.4 seconds	4	32
16	4301.0	7	0.23 seconds	3	38
17	4244.0	7	0.33 seconds	4	32
18	4218.0	7	0.32 seconds	1	33
19	4189.0	8	0.48 seconds	1	31
20	4156.0	9	0.41 seconds	1	24
21	4086.0	10	0.31 seconds	1	31
22	4039.0	8	0.27 seconds	1	23
23	4008.0	8	0.25 seconds	1	26
24	4017.0	9	0.37 seconds	1	23
25	3996.0	8	0.32 seconds	1	21
26	3922.0	9	0.41 seconds	1	23
27	3891.0	8	0.35 seconds	1	22
28	3893.0	7	0.26 seconds	1	23
29	3856.0	8	0.4 seconds	1	21
30	3831.0	11	0.54 seconds	1	22
100	2296.0	3	0.63 seconds	1	9

Table B.9: Simple K Means using Euclidean Distance, max iterations 500, 5 seeds and all categories without demographic data

Number of clusters	**sum of squared errors**	**Iterations used**	**Time to build model[sec]**	**Minimum number of instances in one cluster**	**Maximum number of instances in one cluster**
2	5858.0	4	0.06 seconds	93	123
3	5598.0	10	0.3 seconds	52	110
4	5341.0	9	0.16 seconds	42	73
5	5185.0	14	0.3 seconds	29	67
6	5137.0	12	0.22 seconds	26	52
7	5099.0	9	0.35 seconds	17	57
8	5019.0	12	0.22 seconds	6	59
9	4924.0	8	0.17 seconds	6	43
10	4811.0	8	0.17 seconds	7	37
11	4783.0	9	0.29 seconds	6	35
12	4690.0	11	0.6 seconds	5	35
13	4702.0	7	0.22 seconds	4	33
14	4591.0	10	0.39 seconds	5	28
15	4564.0	12	0.43 seconds	4	31
16	4551.0	8	0.28 seconds	3	33
17	4540.0	9	0.35 seconds	3	30
18	4443.0	9	0.33 seconds	1	23
19	4406.0	10	0.35 seconds	1	26
20	4361.0	7	0.39 seconds	1	24
21	4313.0	9	0.47 seconds	2	29
22	4286.0	8	0.46 seconds	1	23
23	4279.0	7	0.46 seconds	1	25
24	4208.0	12	0.55 seconds	1	17
25	4187.0	7	0.37 seconds	1	17
26	4157.0	7	0.5 seconds	1	17
27	4126.0	8	0.48 seconds	1	15
28	4124.0	7	0.31 seconds	1	19
29	4080.0	7	0.39 seconds	1	18
30	4064.0	7	0.49 seconds	1	18
100	2433.0	4	0.7 seconds	1	11

Table B.10: Simple K Means using Euclidean Distance, max iterations 500, 5 seeds and all categories with demographic data

WEKA-dataset

```
@relation userprofiles

@attribute Sex ’Mann’, ’Kvinne’, ’Annet’

@attribute age ’0-9’, ’10-19’, ’20-29’, ’30-39’, ’40-49’, ’50-59’, ’60-69’, ’70-79’, ’80-89’,
’90-99’

@attribute FilmAction ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmAdventure ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmAnimation ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmBiography ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmComdey ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmCrime ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmDocumentary ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmDrama ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmFamily ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmFantasy ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmHistory ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmHorror ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmMusical ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmMystery ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmRomace ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmSciFi ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmSport ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmThirller ’Liker ikke’, ’Neutral’, ’Liker’

@attribute FilmWar ’Liker ikke’, ’Neutral’, ’Liker’

...

...

...

@attribute ResturantScandinavian ’Liker ikke’, ’Neutral’, ’Liker’

@attribute ResturantGeneral ’Liker ikke’, ’Neutral’, ’Liker’
```

@attribute ResturantItalian 'Liker ikke', 'Neutral', 'Liker'

@attribute ResturantSushi 'Liker ikke', 'Neutral', 'Liker'

@attribute ResturantBBQ 'Liker ikke', 'Neutral', 'Liker'

@attribute ResturantIndian 'Liker ikke', 'Neutral', 'Liker'

@attribute ResturantMexican 'Liker ikke', 'Neutral', 'Liker'

@attribute ResturantVegetarian 'Liker ikke', 'Neutral', 'Liker'

@attribute ResturantSteakHouse 'Liker ikke', 'Neutral', 'Liker'

@attribute ResturantTapas 'Liker ikke', 'Neutral', 'Liker'

@attribute ResturantOriental 'Liker ikke', 'Neutral', 'Liker'

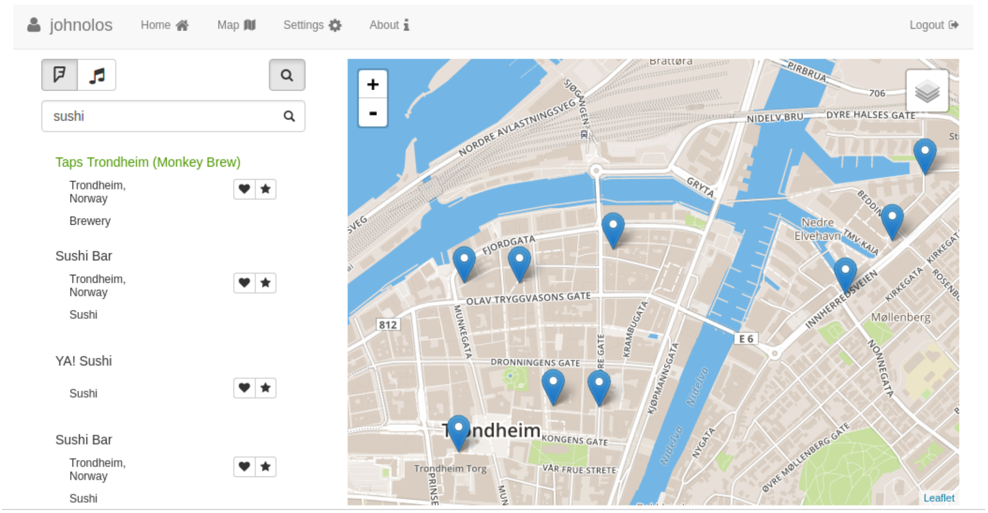
@data

'Mann', '30-39', 'Liker', 'Liker', 'Liker', 'Neutral', 'Liker', 'Liker', 'Neutral', 'Neutral', 'Liker', 'Neutral', 'Neutral', 'Liker ikke', 'Liker ikke', 'Neutral', 'Neutral', 'Liker', 'Liker', 'Neutral', 'Liker', 'Neutral', 'Liker ikke', 'Neutral', 'Liker', 'Neutral', 'Liker', 'Liker', 'Neutral', 'Liker ikke', 'Neutral', 'Liker ikke', 'Liker', 'Liker', 'Liker', 'Neutral', 'Liker ikke', 'Liker ikke', 'Neutral', 'Liker', 'Liker ikke', 'Liker', 'Liker', 'Liker', 'Liker', 'Liker', 'Neutral', 'Neutral', 'Neutral', 'Liker', 'Liker', 'Neutral', 'Neutral', 'Liker', 'Liker', 'Liker ikke', 'Liker', 'Neutral', 'Neutral', 'Liker ikke', 'Liker', 'Neutral', 'Liker'

'Mann', '20-29', 'Liker', 'Liker', 'Liker', 'Liker ikke', 'Liker', 'Liker', 'Liker ikke', 'Liker', 'Neutral', 'Liker', 'Liker', 'Liker', 'Neutral', 'Liker', 'Liker', 'Liker', 'Liker ikke', 'Liker', 'Liker ikke', 'Neutral', 'Liker', 'Neutral', 'Liker', 'Liker ikke', 'Liker', 'Liker', 'Liker', 'Liker ikke', 'Liker ikke', 'Liker ikke', 'Liker ikke', 'Liker', 'Liker', 'Neutral', 'Liker ikke', 'Liker ikke', 'Liker', 'Neutral', 'Neutral', 'Liker', 'Liker', 'Liker', 'Liker', 'Liker ikke', 'Liker', 'Liker', 'Liker', 'Liker', 'Liker', 'Neutral', 'Liker ikke', 'Liker', 'Neutral', 'Liker ikke', 'Liker', 'Liker', 'Liker', 'Liker ikke', 'Liker', 'Liker', 'Liker'

Appendix C
Implementation

C.1 Front-end Screenshots



sushi

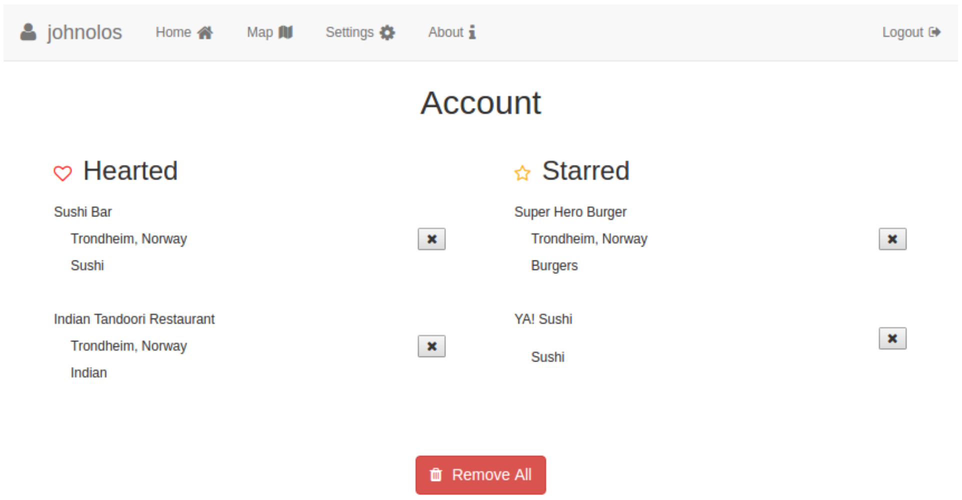
Taps Trondheim (Monkey Brew)
Trondheim, Norway
Brewery

Sushi Bar
Trondheim, Norway
Sushi

YA! Sushi
Sushi

Sushi Bar
Trondheim, Norway
Sushi

Figure C.1: Overview of recommendation page



johnolos Home Map Settings About Logout

Account

Hearted

Sushi Bar
Trondheim, Norway
Sushi

Indian Tandoori Restaurant
Trondheim, Norway
Indian

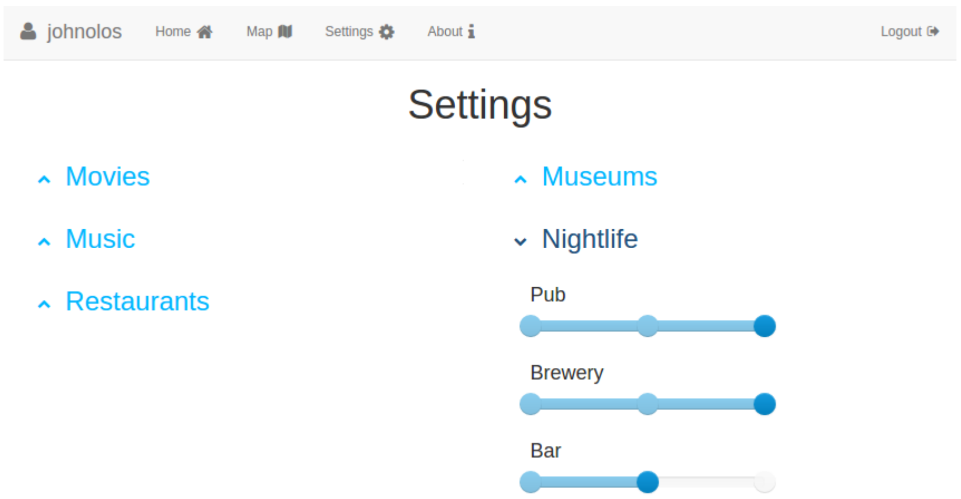
Starred

Super Hero Burger
Trondheim, Norway
Burgers

YA! Sushi
Sushi

Remove All

Figure C.2: Overview of profile



johnolos Home Map Settings About Logout

Settings

Movies

Music

Restaurants

Museums

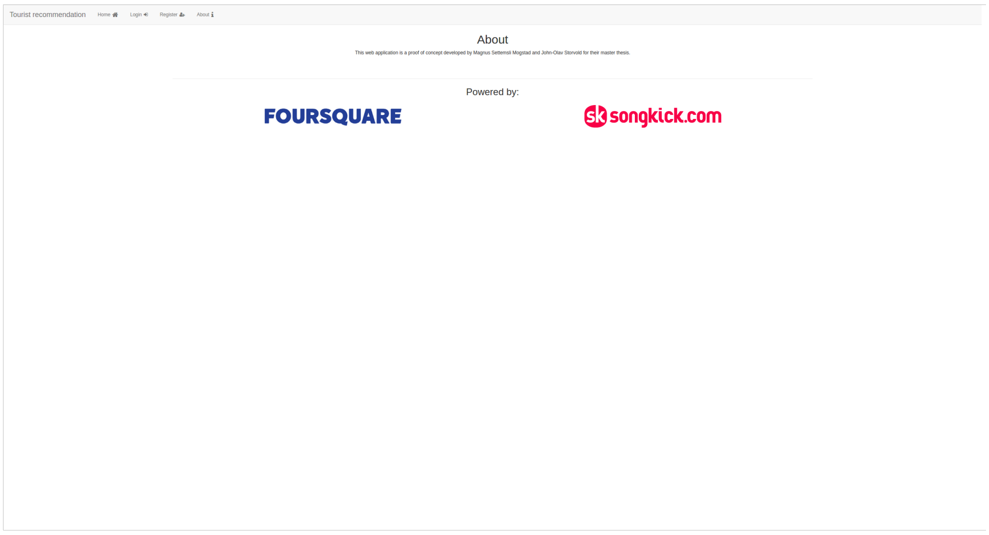
Nightlife

Pub

Brewery

Bar

Figure C.3: Overview of settings



About

This web application is a proof of concept developed by Magnus Settemsli Mogstad and John-Olav Storvold for their master thesis.

Powered by:

FOURSQUARE

songkick.com

Figure C.4: About service

C.2 Server Configuration

For our user test and system test and we hosted our web application on DigitalOcean. DigitalOcean offers cheap, flexible and reliable virtual machines in which you quickly can set up and host a web server. You get root access to a Linux distro of your choosing, and you can install and run your virtual machine with any software you would like. We started out with a fresh Ubuntu 14.04 LTS. After we set up iptables, a light-weight firewall that comes with the Linux kernel. We specified that TCP connections on port 22, 80 and 4000 were allowed. Further, we installed Nginx, Redis, and MongoDB. Nginx is the only topic of those three we have not briefly touched and to be complete we will go into detail of what Nginx does for us.

Nginx[11] is a HTTP, reverse proxy server and a generic TCP/User Datagram Protocol (UDP) proxy server. It excels at server static index files and provides a robust server overall for incoming connections. Node.js, even being great having the capability to handling incoming HTTP/S connections, has to step aside for Nginx when it comes to performance as it can receive 1.9 times as many requests per seconds compared to Node.js averaging at 1.608 requests/sec[19]. In our case, our Nginx strictly acts as a reverse proxy to route Hypertext Transfer Protocol Secure (HTTPS) traffic to our Node.js instance relieving our Node.js of encryption responsibilities which can be a heavy burden.

Moving on, we install our npm modules dependencies and bower components de-

```
#!/bin/sh
export PATH=/usr/local/bin:$PATH
export NODE_ENV=production
forever stopall
sleep 2
forever start --sourceDir /home/john/webapps/tourism/bin/ www
```

Listing C.1: Unix script to instantiate web server

pendencies by running `npm install` and `bower install`. Lastly, we finish off the installation process by installing our python package dependencies for numpy, pymongo, scikit-learn and scipy.

Forever.js and cron

We have taken precautions to ensure that our server always attempts to keep the instance running. We run the Node.js instance with forever.js. Forever.js always strives to run a given script continuously. In the case that runtime errors occur, forever.js logs the appropriate stack traces for the instance id and fire up a new instance to take the crashed instance's place. In the event that the server itself goes down, we have a cronjob in place to run a script that re-instantiates the web server process after the server has come back up.

Appendix D
Evaluation

D.1 User Test

Welcome to user testing of the tourist attraction recommendation system, you can at anytime during the test withdraw.

Tasks you are going to execute:

1. Open the web page https://198.211.120.223:5443/
2. Click "Register" and enter your details.

– We don't ask for or store sensitive information none of which can be traced back to you. Please choose a username that can't identify you.

3. You will be redirected to a settings page. Please edit your preferences accordingly to your taste between a rating from dislike, neutral or like.
4. Head over to "map" in the navigation bar. "Map" is also where recommendations will eventually show up.

– Keep in mind that what you're looking at on the map is also the area location you're searching through. Feel free to head over to a different city should you chose to do so.

5. Using Foursquare (F) as your source of data, find some attractions in the following categories:

– *Friendly reminder: Heart means you'd like to visit, and the star means you have visited before. Try to find places you have been before and star them.*
– Museums
– Restaurants (types: Indian, Steakhouse, Sushi, Kebab, etc.)

- Feel free to search for something else you'd like.

6. Using Songkick (note) as your source of data, do the following:

- Click on the local data source (map pin), find something that seems interesting to you and heart or star them.

- Click on (note means global) and search for your favorite artists. Heart or star as you feel appropriate.

◦ For example Muse, Kygo or Bruce Springsteen.

7. Click on your username in the navigation bar. Please remove some entities from hearted or starred.

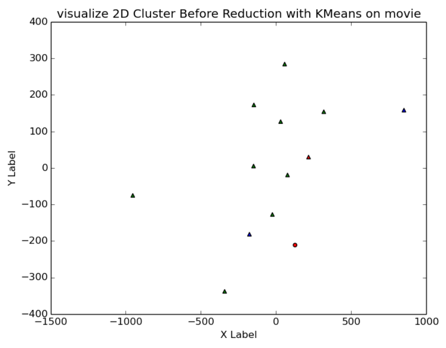
8. Head back over to "map". Recommendations should now have an effect, and when you search, recommended activities and places should be present in the results with green text.

9. Log out of the system.

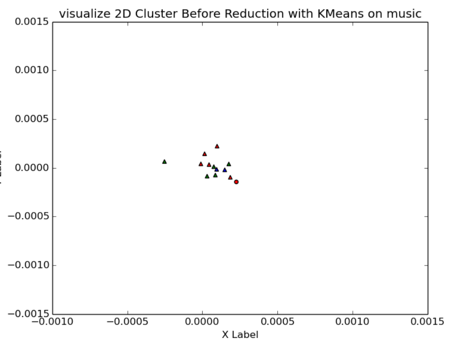
10. Go to the survey and answer the questionnaire.

Thank you for participating.

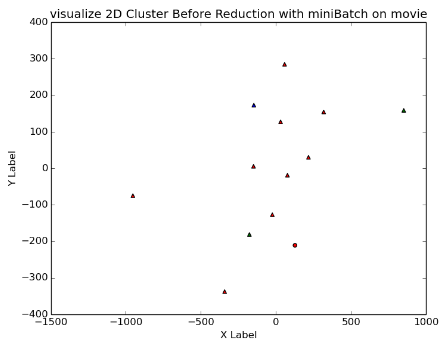
D.2 Cluster Visualization



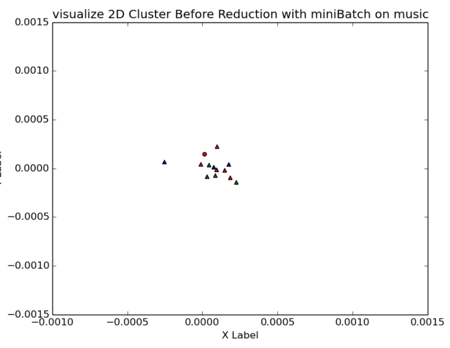
(a) K-Means



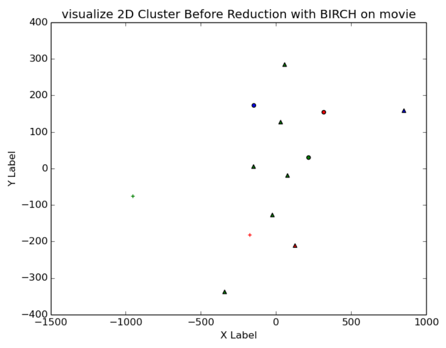
(b) K-Mean



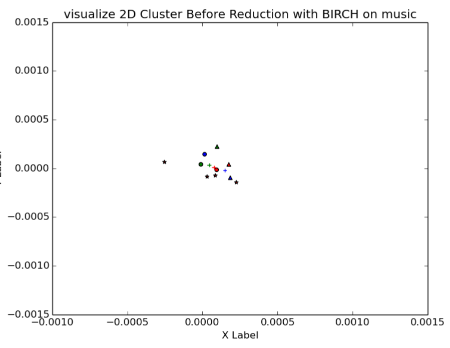
(c) Mini Batch K-Means



(d) Mini Batch K-Means

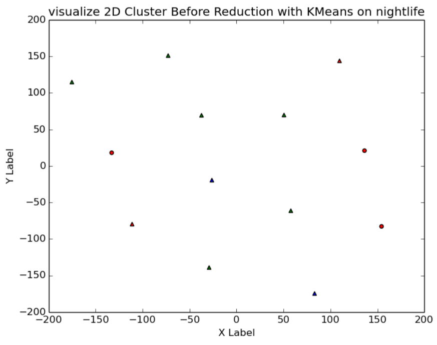


(e) BIRCH

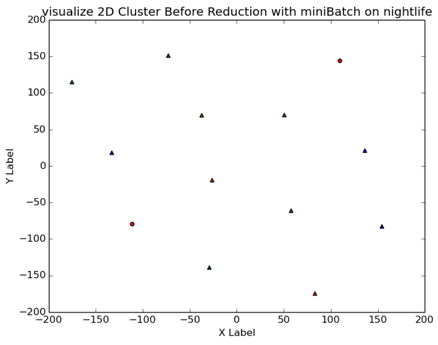


(f) BIRCH

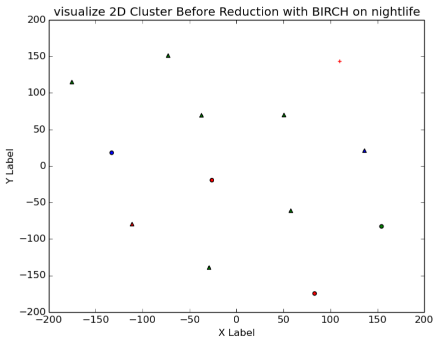
Figure D.1: Visualization of clusters using K-Means, Mini Batch K-Means and BIRCH (a),(c) and (e) is movie category and (b),(d) and (f) is music category



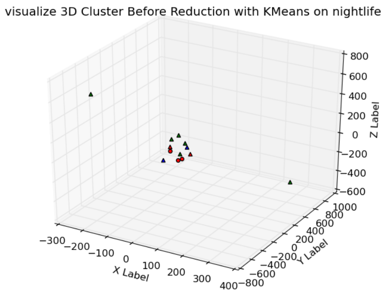
(a) K-Means



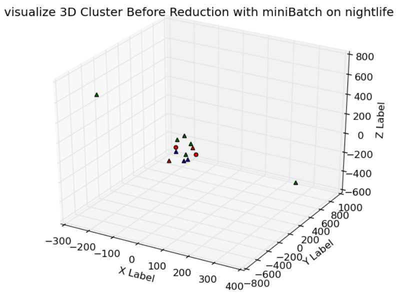
(b) Mini Batch K-Means

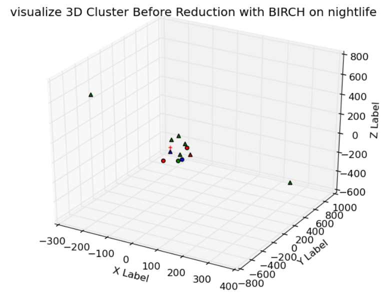


(c) BIRCH



(d) K-Means

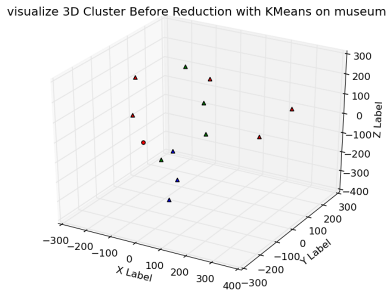




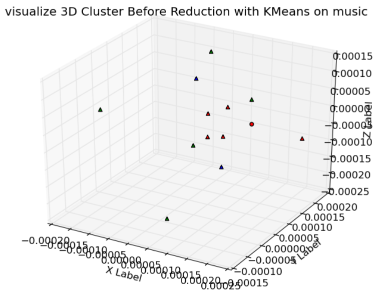
(e) Mini Batch K-Means

(f) BIRCH

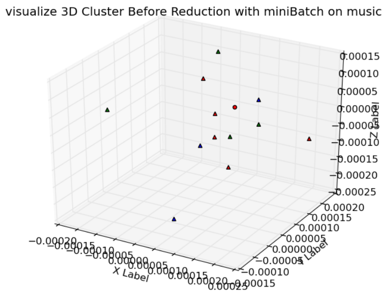
Figure D.2: Visualization of clusters in category nightlife using K-Means, Mini Batch K-Means and BIRCH



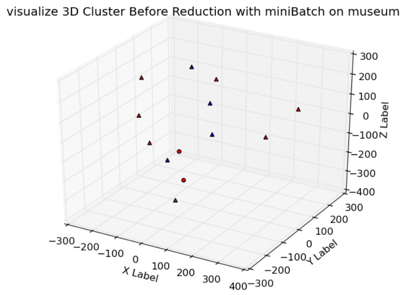
(a) K-Means



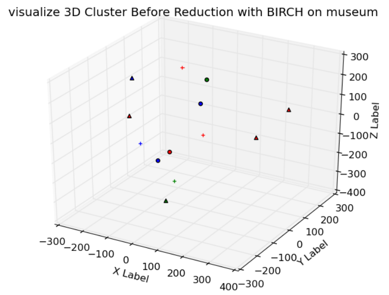
(b) K-Means



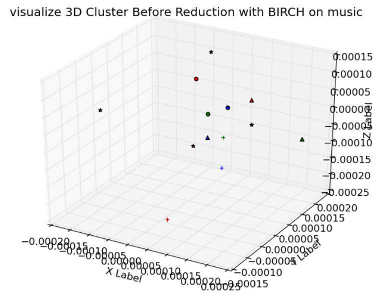
(c) Mini Batch K-Means



(d) Mini Batch K-Means

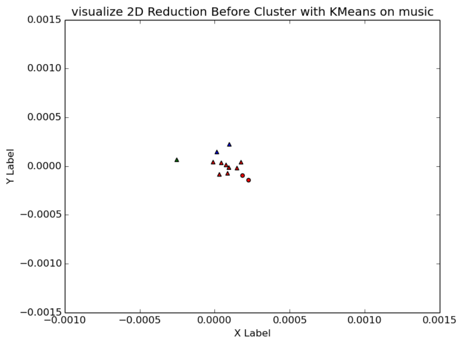


(e) BIRCH

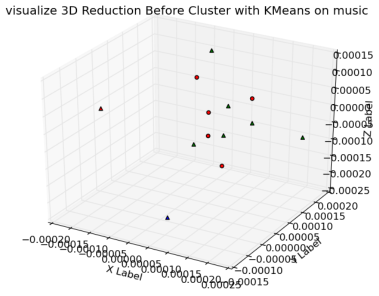


(f) BIRCH

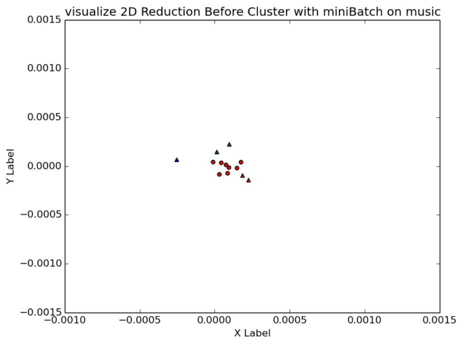
Figure D.3: Visualization of clusters in music and museum categories using K-Means, Mini Batch K-Means and BIRCH



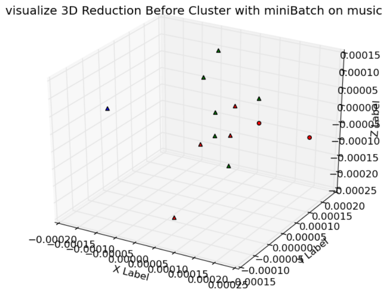
(a) K-Means



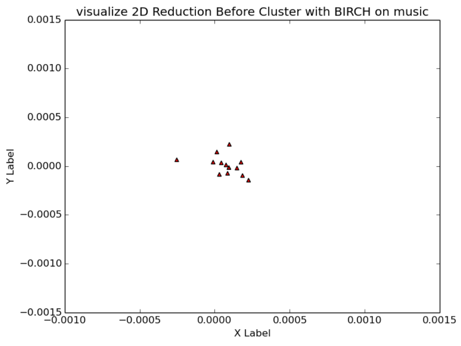
(b) K-Means



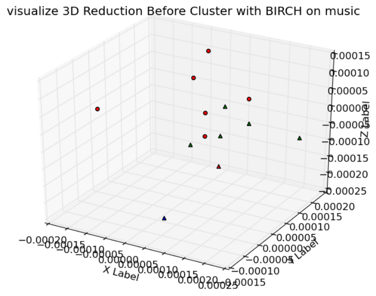
(c) Mini Batch K-Means



(d) Mini Batch K-Means



(e) BIRCH



(f) BIRCH

Figure D.4: Visualization of clusters in category music using K-Means, Mini Batch K-Means and BIRCH on data which has been reduced to either 2 or 3 dimensions before clustering