Property Recommendation using Content Based Filtering and Empirical Data Analysis

Snehal Nair

A thesis presented for the degree of
Masters of Science

Data Science Institute
Lancaster University
Lancaster
24.10.2017

Abstract

Recommender system is an ubiquitous tool that has recently gained prominence in the real estates where the goal is to recommend relevant properties to users to buy or to rent. In this thesis, we develop a suitable recommender system for Guide2Property, a Ghent based start-up property recommendation company.

Existing approaches of recommender system are divided in two categories: collaborative filtering and content based filtering. Both these categories encompasses a diverse set of machine learning tools. We observe that collaborative filtering depends extensively on the amount of users interactions, and thus, not suitable for our dataset.

Content based filtering approaches, despite their own limitations, is more suitable for our dataset. We review two existing content based filtering approaches and propose a new one based on the idea of empirical fuzzy membership function.

The proposed approach is motivated from the idea of fuzzy logic which quantifies a concept, e.g., 'high', 'medium' or 'large', through a membership function. In our case, the membership function quantifies the user's intent from the properties he/she has liked and disliked. The proposed approach assigns a membership value between 0 to 1 to each property which is intuitively interpretable.

We apply the proposed approach to recommend relevant properties, and observe that it performs well compared to existing methods both quantitatively and qualitatively.

Acknowledgements

I would like to thank Prof. Angelov Plamen for his help and supervision with this project, Wendy Geeraert to offer this project to me, the technical person at Guide2property, Marteen Belmans to provide me insights into the data and Prof. Debbie Costain and Prof. Simon Tomlinson for their constant support with the dissertation program.

I would also like to thank my sister, Swapna Nair, my brother-in-law, my mother, father for their never ending love, support and care. My little niece, Veda and my friends for the source of joy, they are and my husband, Sohan Seth for his self-less support, mentoring and love.

Contents

1 Introduction

Co-libry[1] is a real-estate search engine start-up venture under Guide2Property in Belgium, which aims at making faster and better recommendation to its user to find a perfect home. The current recommendation system used by Co-libry is user-search based or rule (constraint) based method on geographic location, price, property space, financial factors, information on the surrounding area (e.g., health, shopping, school, parks, etc.), and other miscellaneous factors, e.g., the mortgage calculator. It is a purely knowledge-based recommendation coupled with location-based search optimization (See 1.1.1) for more details).

1.1 Guide2Property

1.1.1 Recommendation method

At Guide2Property the standard search engine is used for recommending relevant properties[2]. A standard search engine builds queries based on the user input on the website and executes it in the database to bring out the search results to the user. On the website, the user first enters the input for, e.g., properties for sale or properties for rent, and this creates the first base query. Based on the next input given, e.g. the city name, another query is created. The query is thus updated for every user input, and the final query is executed on the property database to provide recommendation based on the search results. To summarize, if a user enters the information property on sale, 3 bedroom, in the city of Ghent, then the respective query is created and the standard search engine figures out which properties should be listed for the users request for 3-bedroom house in Ghent for sale.

While this approach is fairly simple, there are few parameters that makes the current search engine powerful. For example, a user can enter a 'Point of Interest' on the website. The user can then specify how far he wants to live from his point of interest, say, within 20 minutes distance by train. The query thus updates automatically to include this location based information. Another striking feature is that a user can enter an 'environment score' which is a rating from 1-3 to denote how important these factors, e.g. health, green, shops, train, etc. are to the user. Based on the open Google Maps data, the query updates to include these parameters in the property search. This system works well to provide recommendations based on the user's request. A registered user can also save the requests,

[1] `https://co-libry.com`

[2] Personal communication with Marteen Belmans.

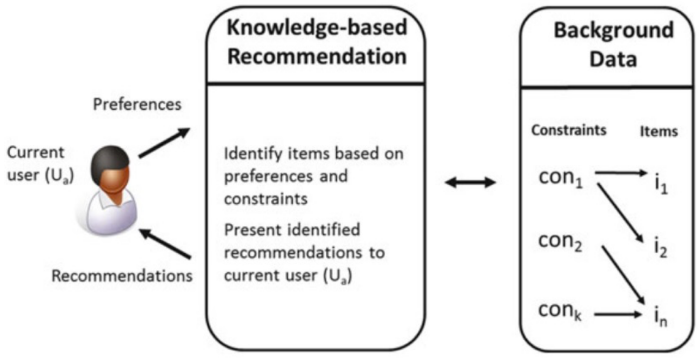


Figure 1.1: Knowledge-based recommendation (KBR) dataflow: users are entering their preferences and receive recommendations based on the interpretation of a set of rules (constraints)
source: From the paper "Personalized views of personalization" by D. Riecken

and search it again on the website, each time he/she revisits. The current system follows a recommendation approach called Knowledge-based recommendation as shown in the Figure 1.1 relies on the background data like, a set of rules (constraints) or similarity metrics and a set of items. Depending on the given user requirements, rules (constraints) describe which items have to be recommended.

This approach, however, lacks personalisation since it is purely constraint driven than intent driven. Traditional recommender systems method, on the other hand, are intent driven in the sense that they recommend items based on the users feedback on other items rather than his/her search queries.

As a side note, a form of personalisation can be implanted in the search based approach. For example, Yuan et al. [2013] proposes a user-oriented recommendation system for real estate websites via a combination of case-based reasoning (CBR) and an ontological structure. CBR is a problem solving methodology that addresses a new problem by retrieving similar solved problems and reusing the results for solving current at hand. They used the customer requirements/constraints in a house search to construct an ontology framework. To begin with, a query is sent based on user's input. The query analyser then searches the database, and display related solution. Once the users narrow down their requirements, the similarity measurements compare the user's query and database cases in order to provide recommendations. Thus, the ontology framework gets narrowed down based on user preference. The case are built and stored to use it for recommendation when similar case is queried by the user again. We reviewed this work in the 2.6.3. This approach, however, is beyond the scope of this thesis.

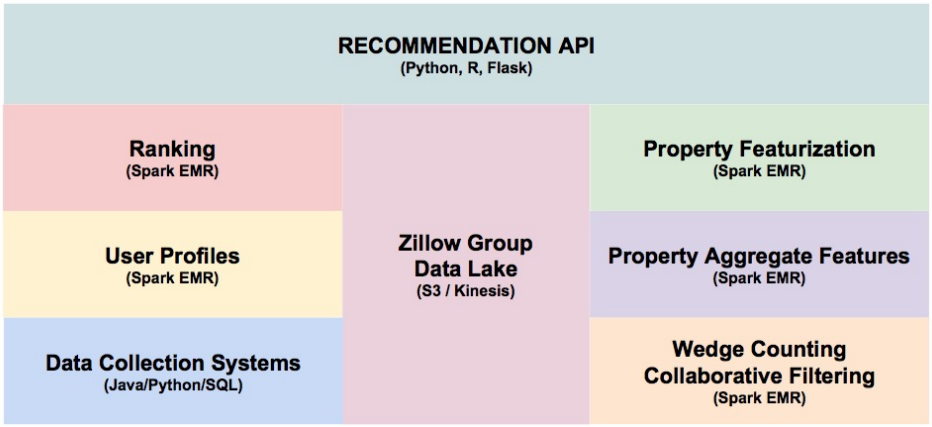


Figure 1.2: Zillow/Trulia Recommendation Architechture

1.1.2 Architecture

This section briefly describes the architecture in the current system which includes database, scalability and centralized data store. The company is currently using two database platforms like SQL and MongoDB where smaller datasets like all registered users , request for contacting a seller are stored in the SQL database and big datasets like properties, user-search profiles, location-based data and in-house analytics (like user-clicks) and feeds from scraping other websites for new-listings is stored in MongoDB. Both the databases, including webserver are running on Microsoft Azure.

We also reviewed the architecture used by topmost real estate websites in the world, Zillow, Trulia. The Figure 1.2 gives a snapshot of the architecture at Zillow. Trulia followed the same architecture as it was acquired by Zillow.

The recommender system at Zillow and Trulia is based on the same model 'Wedge Counting'. Wedge Counting methodology for recommendation was published by Joseph S. Kong et all []. The process followed for recommendation is : Predict homes (likes/dislikes) based on the behaviour of similar users. Wedge count method is used for all user-property pairs to predict user likes/dislikes. Based on the like/dislike counts, the user-item preference is classified using Gradient Boosting Classifier (sklearn). In the next step, user profiling is carried out based on collaborative-filtering method. The output of these two step is a property matrix which is ranked based on the final scores received.

1.2 Project Overview

Based on the background information gathered, in the above sections, property search in general is not based on hard decisions/constraints. For example, although a user inserts '2

bedroom property under 700£' as a requirement, he/she might be interested in '1 bedroom property at 500£' or a '2 bedroom property at 800£' given other criteria is met by the property. For example, these properties might be close to the city center, might be situated near convenient stores, school or leisure centers etc. The main problem associated with the current recommender system is that it fails to surprise the user with 'wonderful discoveries', instead only recommends the 'obvious' properties, based on the user's constraint search.

We suggest moving from *constraint based search to intent based search* where similar properties are found by modelling the user's personality through his/her likes and dislikes. This approach, by design, invokes personalisation. Personalisation is a necessary step toward building customer loyalty, where an enterprise builds a meaningful one-to-one relationship with its customer base through understanding the needs of each individual, and helping satisfy a goal that efficiently and knowledgeably addresses each individual's need in a given context. Therefore, our objective is to build an improved recommendation engine that looks beyond the constraint based search, and recommend properties based on the user's likes and dislikes.

Recommendation systems that model the user's intent have been extensively studied in the literature. For example,

1. Learning different user personalities to cross-recommend items liked by the users of similar personality. The personalities can either be learned based on their online behaviour of rating items or by using their demographic information. This is a generalised recommendation technique for segmented population of users based on their personalities.

2. Learning similarities between the items selected by the user and other items to recommend items with high similarities. This can be done in two different ways, the item similarities can be learned either by comparing user reviews of the items, or by comparing the attributes of the item (features). This approach technique is more *personalised* as it looks for similar items to the items selected by the user.

Both these approaches have their pros and cons. For example, any approach which relies on the user's rating requires a large number of users to rate a large number of items. Although such approaches are sensible and quite effective given sufficient data, it might not be suitable for a start-up organization with relatively small registered users. On the other hand, approaches that measure similarities between items may suffer from choosing the right similarity metric, but it can be applied to a small user base even if each user has only rated very few items.

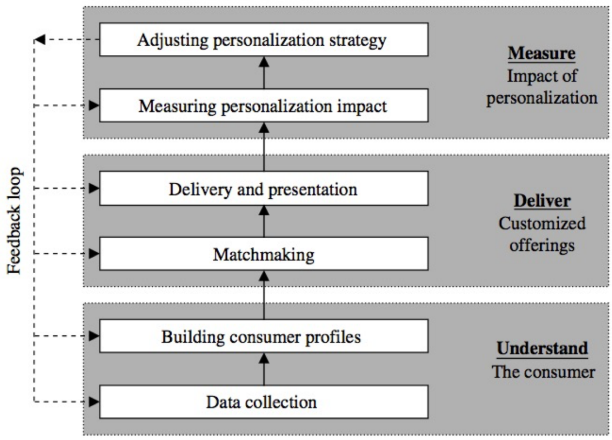


Figure 1.3: Personalisation process (UDM model). source: From the paper 'Personalisation technologies: A process-oriented perspective' by G. Adomavicius et. al.

We consider the latter approach and propose a new method based on the idea of *empirical data analysis* (EDA). EDA is a solely data-driven machine learning tool that is free of user-specific prior assumption [Angelov et al., 2016]. Recently this idea have been extended to fuzzy logic where the membership functions are not specified a priori by the user, but instead estimated in a data-driven fashion. These are referred to as the *empirical membership function*. This idea is particularly useful in our context since the user's intent can be represented as a membership function, and properties with high memberships can be recommended to the user. We follow this strategy to develop our novel recommendation algorithm, and show that it performs well on our dataset compared to other methods.

1.3 Research Aim

With this research project, the client is looking at ways to build user loyalty, acquire more users and improve conversion rate by personalising it for it's users. Personalisation is about building customer loyalty by building a meaningful one-to-one relationship through understanding the needs of each individual, and helping satisfy a goal that efficiently and knowledgeably addresses each individual's need in a given context [3].

As argued by [Adomavicius and Tuzhilin, 2005a] , personalisation in its general form is an iterative process consisting of several stages that are integrated together into one system. In particular, Adomavicius and Tuzhilin [6] propose the following understand-deliver-measure (UDM) framework (see Figure 1.3).

1.4 Organization

In §2, we provide a detailed overview of the existing literature on recommender systems. The existing methods can be conceptually classified into two broad categories, i.e., the Content Based Filtering (CBF) (§2.2.1) and the Collaborative Filtering (CF) (§2.2.2). Both these ideas encompass many machine learning tools, and we review a few of them. We discuss the limitations of CF and CBF approaches in §2.3. In §2.6, we focus on recommender systems for real estates. Here, we discuss a popular CF based approach, i.e., square counting in §2.6.1, and in §2.6.2, we discuss the idea of fuzzy logic based recommendation systems that we develop in this thesis.

In §3, we discuss the necessary data processing steps for cleaning the raw data for missing values, outliers, etc. along with suitable feature transformations. This results in two data structures that we will use throughout this thesis; the 130,708 $\times$ 19 *property data* matrix, and the array of properties lists both liked and disliked by 15 users (with 5 users liked/disliked >2 properties). First, we discuss three CB methods, namely, neighborhood based, model based, and square counting based in §3.2, for recommending properties. We observe that these methods fail in our problem due to lack of data. Next, we discuss three CBF method, namely regression based, similarity based, and the proposed approach, i.e., the fuzzy rule based. Since these methods do not rely on the interactions among users, e.g, liking the same properties, they work well.

We discuss the performance of the CBF approached in §4. One of the most widely used method for evaluating recommender systems is the area under the precision recall curve. However, we observe that it is difficult to assess the systems using such metric due to lack of testing samples since each user only likes very few properties. However, nonetheless, we observe that the proposed method performs better than the two baseline methods in §4. We further analyse the result qualitatively by observing the recommended properties in two dimensions in §4.2.

In §5, we summarize the proposed approach, address its advantages, discuss its potential drawbacks, and suggest further improvements. In §6, we revisit our research aims, and discuss our approach in the context.

The following table details the notations used in the thesis.

Notation	Comment
i	Item (property)
n	Number of items (properties)
j	Feature
m	Number of features
u	User
t	Number of users
p	Prototype
k	Number of prototypes
θ	User profile vector
Θ	Matrix of user profile
$\mathbf{x}$	Property feature vector
X	Matrix of property features
$\mathbf{p}$	Prototype property feature vector
P	Matrix of prototype property features
Y	Utility matrix of like/dislike

Table 1.1: Mathematical notations

2 State-of-the-Art Approaches

2.1 What is a Recommender System (RS)?

Information explosion started with the *dot-com bubble* during 1992-2002 [Goodnight and Green, 2010] when a variety of e-business services, commonly referred to as *dot-coms* were founded. These services offered their users a vast choices of products on the internet. Choosing the right product among *too many* options soon became a challenge for the users, and this innate difficulty paved the way for Recommender Systems (RS). RS is an information filtering tool that seeks to predict which product a user will like, and based on that, recommends *a few* products to the users. For example, Amazon can recommend new shopping items to buy, Netflix can recommend new movies to watch, and Google can recommend news that a user might be interested in.

During the 1990s, Lee et al. [1998] published their work on *assistant agent* as part of the CiteSeer project where the authors addressed the issue of finding relevant web-based research publications. In their paper, they described a assistant agent that tries to automate the process of finding relevant articles and suggesting them to the users. The authors found similarity between two publications by assessing the distance between their respective term frequency - inverse document frequency (TFIDF) vectors. While the term recommender system was coined by Resnick and Varian [1997], the term assistant agent used by Lee et al. [1998] is synonymous to the recommender system. The first recommender system, Tapestry at Xerox Palo Alto Research Center, was built by Goldberg et al. [1992] who coined the term *collaborative filtering* when they proposed that “information filtering can be more effective when humans are involved in the filtering process".

To understand the concept of recommender systems, let us look at an example. Table 2.1 shows the user-item *utility matrix* Y where the value R_{ui} denotes how item i has been rated by user u on a scale of 1-5. The missing entries (shown by **?** in Table 2.1) are items that have not been rated by the respective user. The objective of the recommender system is to predict the ratings for these items. Then the highest rated items can be recommended to the respective users. In real world problems, the utility matrix is expected to be very sparse, as each user only encounters a small fraction of items among the vast pool of options available. In the following sections we review the different approaches to solve this problem.

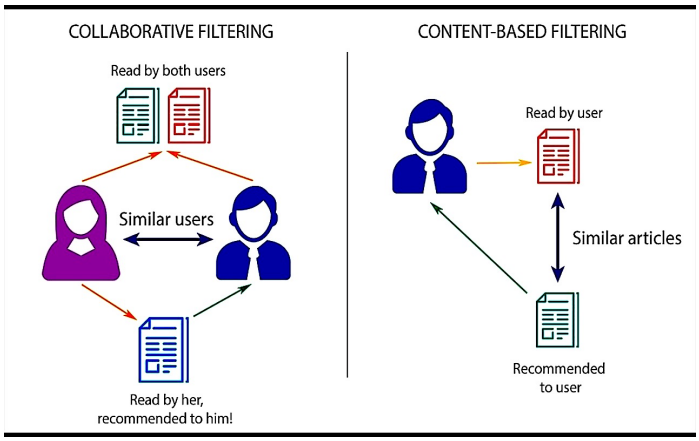


Figure 2.1: (Left) Content-based filtering where similar users are identified based on what they read (liked). **(Right)** Collaborative filtering where similar articles are identified based on what the user read (liked). Source: `http://i65.tinypic.com/2ebah6c.png`

Table 2.1: Utility matrix showing user-item pairwise ratings

	item1	**item2**	**item3**	**item4**
user1	2	5	1	3
user2	4	?	?	1
user3	?	4	2	?
user4	2	4	3	1
user5	1	3	2	?

There has been a considerable amount of work on recommender systems; see for example Herlocker et al. [1999], Adomavicius and Tuzhilin [2005b], Burke [2002] and Ekstrand et al. [2011]. Jannach and Friedrich [2013] and Ricci et al. [2011] published a book on recommender systems. While Fayyad et al. [1996] and Berendt et al. [2002] focused on real-estate web recommendations in their books. We reviewed the work of Beel et al. [2016] for a succinct understanding of this area of research.

2.2 Approaches to Recommendation

The two widely used approaches for building a recommender system are the content-based filtering (CBF) and collaborative filtering (CF), of which CBF is the most widely used [Akhtar and Agarwal]. Figure 2.8 illustrates the concepts of CF and CBF. The primary difference between these two approaches is that CF looks for similar users to recommend items while CBF looks for similar contents to recommend items. A detailed description of

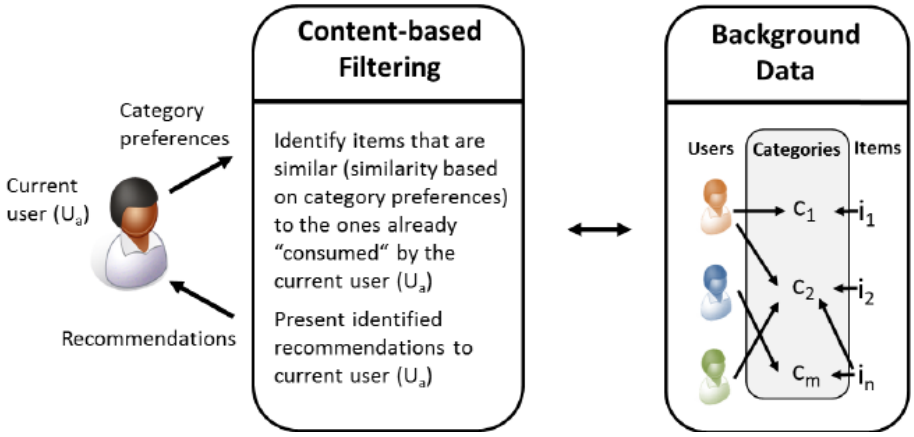


Figure 2.2: Content Based Filtering
source: researchgate by Alexander Felfernig

the two methods is provided in the next sections.

2.2.1 Content-based Filtering

The main idea behind CBF is to recommend items similar to the items previously liked by the user. For example, if the user have rated some items in the past, then these items are used for *user-modeling* where the user's interests are quantified. Traditionally, the item i is represented by a *feature vector* $\mathbf{x}_i$, which can be boolean or real valued, and the user is represented by a weight vector θ_u of same dimension. Given a new item $\mathbf{x}^*$, represented in the same feature vector space, the likeliness, e.g., rating of the item is predicted using the user model. Figure 2.2 shows the how a content based model operates.

This can be achieved in two different ways:

- Predicting ratings using parametric models like regression or logistic regression for multiple ratings and binary ratings respectively based on the previous ratings.
- Similarity based techniques using distance measures to find similar items to the items liked by the user based on item features.

CB can be applied even when a strong user-base is not built, as it depends on the item's meta data (features) therefore does not suffer from cold-start problem. However, this also makes it computationally intensive, as similarities between each user and all the items must be computed. Since the recommendations are based on the item similarity to the item that the user already knows about, it leaves no room for serendipity and causes over specialisation. CB also ignores popularity of an item and other users feedbacks. The paper by Giles et al. as part of the CiteSeer project discussed in the section 2.1 is also an example

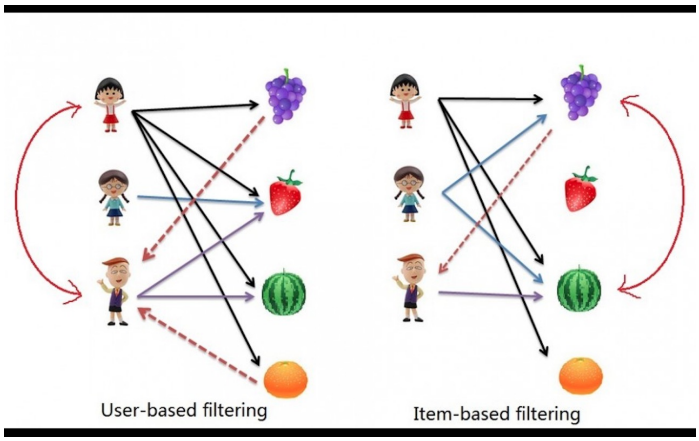


Figure 2.3: (Left) User-User Based CF: Where the similarity between users are identified based on their rating pattern they are also called the neighbourhood users. The items are therefore liked by the users in the neighbourhood are recommended for users in the group. recommend items. **(Right)** Item-Item Based CF: Where the similarity between items are identified based on the ratings given by the users, they are also called the neighbourhood items. source: `http://www.salemmarafi.com/wp-content/uploads/2014/04/collaborativeFiltering-960x540.jpg`

of CB filtering [Lee et al., 1998]. Listed below is most popular content based filtering based recommendation systems used in different fields. This includes recommender sytems like LIBRA (for book recommendation) [Mooney and Roy, 2000b], CBMRS (music) [Kim et al., 2006].

2.2.2 Collaborative Filtering

Collaborative filtering aggregates the past behaviour of all users. It recommends items to a user based on the items liked by another set of users whose likes (and dislikes) are similar to the user under consideration. This approach is also called the *user-user* based CF. Later the *item-item* based CF became popular Ricci et al. [2011], Linden et al. [2003] where to recommend an item to a user, the similarity between items liked by the user and other items are calculated. However, contrary to CBF, here the similarity between items is measured using user ratings of those items, rather than their contents. The Figure 2.3 shows how user-based and item-based CF works. The user-user CF and item-item CF can be achieved by two different ways, i.e., the memory-based (neighbourhood) approach, and the model-based (latent factor models) approach.

1. **Neighbourhood approach**: To recommend items to user u_1 in the user-user based neighborhood approach first a set of users whose likes and dislikes similar to the user u_1 is found using a similarity metrics which captures the intuition that $\text{sim}(u_1, u_2) > \text{sim}(u_1, u_3)$ where user u_1 and u_2 are similar and user u_1 and u_3 are dissimilar. This

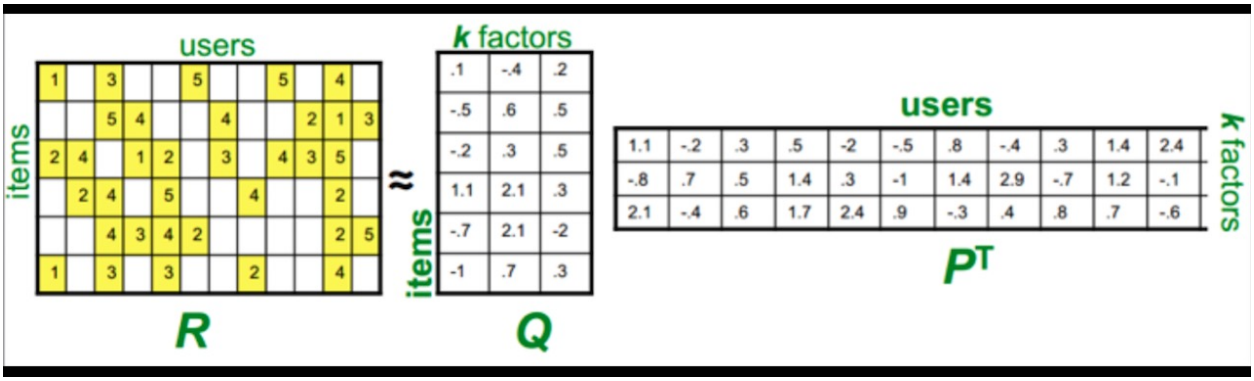


Figure 2.4: The product of the two matrices in the RHS is used to predict the missing ratings in the users-item utility matrix in the LHS. Source: `https://blogdotrichanchordotcom.files.wordpress.com/2015/11/latentfactor.png`

similar user is called the neighbourhood of user u_1.

Neighbourhood approaches are most effective at detecting very localized relationships (neighbours), ignoring other users. But the downsides are that, first, the data gets sparse which hinders scalability, and second, they perform poorly in terms of reducing the RMSE (root-mean-squared-error) compared to other complex methodsLucas et al. [2013]. Neighbourhood approaches were popularized by Herlocker et al. [1999] and Linden et al. [2003]. An automated collaborative filtering system using neighbourhood-based algorithm was first introduced by GroupLens where they provided personalised predictions for Usenet news articles using Pearson correlations to weight user similarity [Miller et al., 2003].

2. **Latent Factor Models:** Latent factor model based collaborative filtering learns the (latent) user and item profiles (both of dimension K) through matrix factorization by minimizing the RMSE (Root Mean Square Error) between the available ratings y and their predicted values $\hat{y}$. Here each item i is associated with a latent (feature) vector $\mathbf{x}_i$, each user u is associated with a latent (profile) vector θ_u, and the rating $\hat{y}_{ui}$ is expressed as

$$\hat{y}_{ui} = \mu + b_u + b_i + \theta_u^\top \mathbf{x}_i$$

where $\mu + b_i + b_u$ is the baseline predictor which includes the user bias b_u and item bias b_i. Figure 2.4 illustrates this process. This approach is useful since a sparse matrix Y is used to estimate the full matrices X and Θ, and then missing entries of Y is replaced with $\Theta^\top X$.

Latent methods deliver prediction accuracy superior to other published CF techniques [Ricci et al., 2011]. It also addresses the sparsity issue faced with other neighbourhood

models in CF. The memory efficiency and ease of implementation via gradient based matrix factorization model (SVD) have made this the method of choice within the Netflix Prize competition. However, the latent factor models are only effective at estimating the association between all items at once but fails to identify strong association among a small set of closely related items. Other notable works in this area are by Koren [2008], Hofmann [2004], Zhang et al. [2005].

2.3 Limitations of the approaches

As discussed above, every approach has its own advantages and disadvantages and an approach has to be selected based on the problem at hand. We briefly discuss the limitations of these approaches.

- **Over-Specialisation**: CBF looks at item-item similarity based on contents of the items liked by the user. This makes the algorithm susceptible to over-specialisation as the similarity metrics will not select an item that is different from the item liked by the user. Thus, users are restricted to receiving recommendations of the items close the items they have know-how over-specialisation which affects serendipity since diversity is one of the aspects a user looks for in recommendation. Genetic algorithm is gaining popularity with content based filtering to handle over-specialisation, where the solutions produced from the initial algorithm are selected using *fitness function* to reproduce results to handle evolution in preferences of the user [**?**].

- **Data-Sparsity**: The ratio of the number of user to the number of items available is always disproportionate, and a user can never go through all the available items to provide his/her ratings. Therefore methods like CF that require a utility matrix (user-item matrix) usually suffers from sparsity issues. Empirical fuzzy rule based system works is independent of the user-user or item-item interaction. It works primarily on the features, therefore does not face sparsity issues.

- **Scalability** Computationally intensive methods can cause scalability issue. Any algorithm should be built with the capability to scale-up based on the growing number of users and items which can cause exponential growth in computations based on the user and item addition. [Papagelis et al., 2005] proposed using incremental updating the new ratings given by the user in CF thus handling the associated computational issues [Papagelis et al., 2005]. Feature selection, instance selection and other dimensionality reduction techniques can also counter scalability problem and provide better

quality recommendations [Yu et al., 2001, Zeng et al., 2003]. Empirical fuzzy rule based system is a content based approach, that can provide better scalability.

- **Cold-Start problem**: Cold-Start problem can arise during addition of a new user or a new item where both do not have history in terms of ratings. Since CF depends on rating history for the user and item, this can affect the recommendations. However, CBF does not depend on rating history, so it is not susceptible to this problem. As discussed in data-sparsity, empirical fuzzy rule based system works is independent of the user-user or item-item interaction therefore is unaffected by Col-Start problem. It can start making recommendations based on the first item.

2.4 Empirical Data Analysis

Empirical data analysis (EDA) is a purely data-driven, parameter free and prior assumption free data analysis framework Angelov et al. [2016]. Unlike the Bayesian analysis framework that requires assuming a parametric density function, or the fuzzy set theory that requires assuming a parametric fuzzy membership function, EDA introduces the concept of *typicality* which resembles the idea of a density function or a membership function, e.g., it takes value between $[0, 1]$, but it is inherently different and purely data-driven. Given a set of samples $x_1, \ldots, x_n$, with unique values $u_1, \ldots, u_m$, and a distance metric $d(x, y)$, the typicality is defined in four stages as follows.

1. Cumulative proximity measures the degree of closeness/similarity of a particular data point u_i to all other data points as follows:

$$\pi(u_i) = \sum_{j=1}^{m} d^2(u_i, u_j).$$

2. Standardized eccentricity measures the degree of a data point u_i being an outlier by measuring its association with the tail of the distribution as follows:

$$\epsilon(u_i) = \frac{2\pi(u_i)}{\frac{1}{m}\sum_{j=1}^{m} \pi(u_i)}.$$

3. Density is inversely proportional to the standardised eccentricity as:

$$\delta(u_i) = \frac{1}{\epsilon(u_i)}.$$

4. Typicality is the normalized data density as follows:

$$\tau(u_i) = \frac{\delta(u_i)}{\sum_{j=1}^{m} \delta(u_j)}.$$

One can also define the multimodal typicality as follows:

$$\tau^{\text{MM}} = \frac{f_i \delta(u_i)}{\sum_{j=1}^{m} f_j \delta(u_j)}$$

where f_j denotes the empirical count of x's and location u_i.

Intuitively, typicality assigns a small value to samples that are at the outskirts of a pool of samples whereas it assigns large values to samples are surrounded by other samples. τ^{MM} follows a number of desired properties similar to density function or membership function, e.g., i) it realises values between [0,1], and ii) it sums up to 1. However, typicality is purely data-driven and does not require prior parametric assumptions, e.g., a Gaussian density function. Additionally, typicality shares the following advantages,

- Typicality takes into the spatial density δ of the data.
- It also takes into account the frequency of occurrence, f, of a certain data sample.
- Estimating typicality does not require any clustering algorithm, thresholds, or any other parameters
- It has a closed analytical form for simple distance metric, e.g., for Euclidean distance metric the density function δ assumes the form of a Cauchy function.

Typicality can be replaced in standard machine learning algorithms that employ density function. For example, in the naïve Bayes' classifier that assumes a Gaussian density function around the mean of the cluster, one can instead use typicality to assign a new sample to a cluster as follows:

$$\text{argmax}_{c=1}^{C} \tau_k^{\text{MM}}(x)$$

where $\tau_k^{\text{MM}}(x)$ is the typicality at x induced by samples in the k-th cluster. This approach seems to perform better than the former approach based on Gaussian density function [Angelov et al., 2016].

Recently Angelov and Gu [2017] have extended this idea to fuzzy logic where the density function is used a empirical membership function.

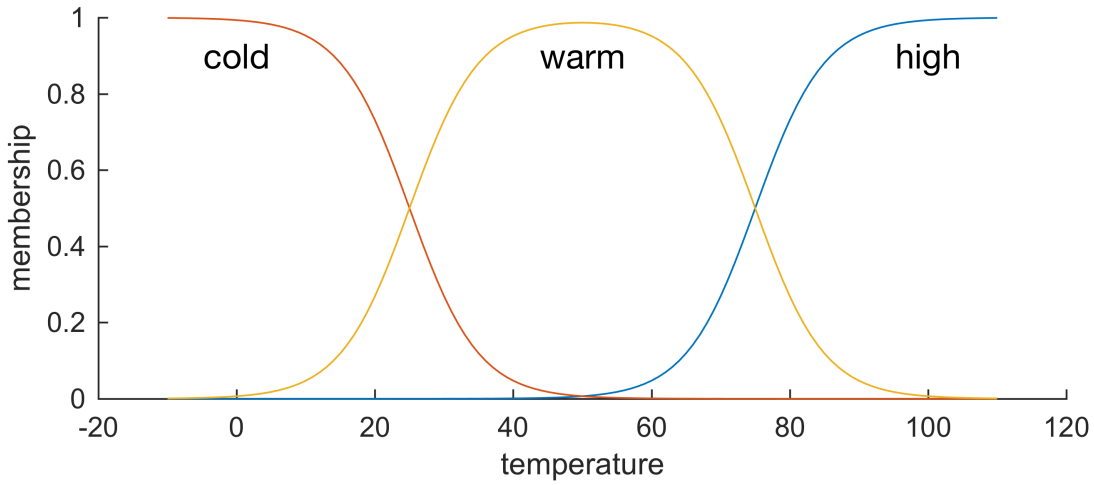


Figure 2.5: Example of fuzzy membership function

2.5 Fuzzy Logic

Fuzzy logic moves beyond the binary world of 'true' and 'false' and introduces the idea of 'degrees of truth', i.e., where a statement might not be true or false, but it can be both true and false with certain degrees. Intuitively, fuzzy logic aims at performing computation using human linguistics. For example, consider the concept of a 'high temperature'. A 'high temperature' does not imply a specific temperature such as 100C, but it is a subjective statement which implies a range of temperatures. Similarly, if we ask whether 22C is a 'high temperature', then the answer is not a binary true or false, but 22C seems to be more of a 'cold temperature' than a 'cold temperature'.

At the core of fuzzy logic is the idea of a membership function. Intuitively, membership function is attributed to a concept, e.g., 'high temperature'. Mathematically, it is a function $m : S \to [0, 1]$ over a set S which assigns a value $m(t)$ to an element $t \in S$ where $m(t)$ is the membership of t in (S, m). For example, 'a high temperature' can be represented by a sigmoid function

$$m_{\text{high}}(t) = \frac{1}{1 + \exp((t - 75)/5)}.$$

Similarly, 'a cold temperature' can be represented by the function

$$m_{\text{cold}}(t) = 1 - \frac{1}{1 + \exp((t - 25)/5)}.$$

Together they imply that certain temperature are more high than cold, i.e., $m_{\text{high}}(t) > m_{\text{cold}}(t)$, but no temperature (in this example) is truly high or truly cold. Figure 2.5 shows the respective membership functions.

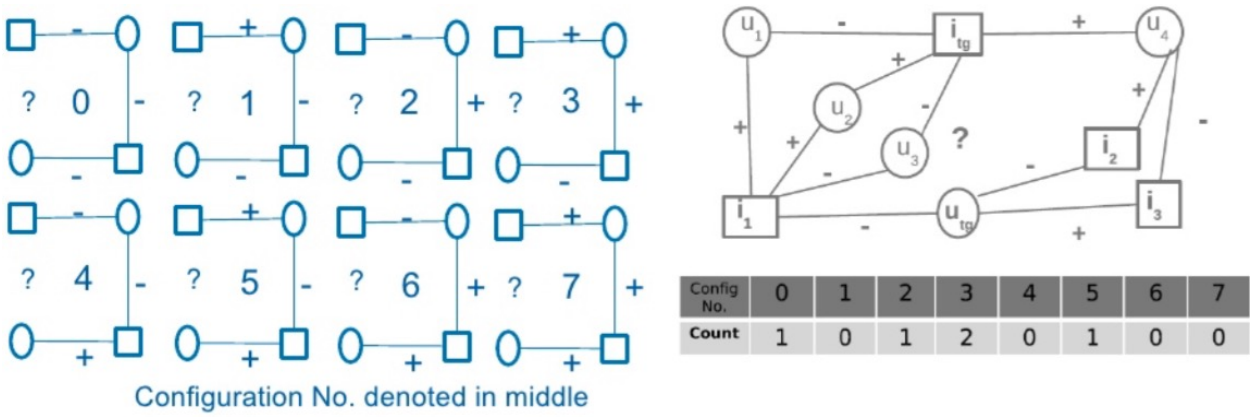


Figure 2.6: (Left) 8 possible like/dislike configurations. **(Right)** An example user-item bipartite graph. source: slideshare presentation by Zillow

The idea of fuzzy membership can be extended to concept beyond simple one dimensional example. In the context of real estates, for example, one can define fuzzy membership functions for a 'small apartment near the city center' or a 'moderately priced family house'. The difficulty in designing these membership functions subjectively, however, is that one does not know what they mean in the context of many other variables, e.g., distance from the airport or if the property has double glazing, in a high dimensional space, or how many membership functions one need. Empirical data analysis offers solution to this problem by defining the empirical membership function.

2.6 RSs for Real Estate

Recommendation systems is gaining popularity with real estates services, for example, Zillow[3] and Trulia[4] are two of the most popular real estate companies that recommend real estates to users. Since Trulia was acquired by Zillow (until it started operating independently again later), both these companies use the same recommender system which uses the *love-hate square counting method*, a CB based approach proposed by Kong et al. [2012] to find relevant properties. Angelov and Gu [2017], on the other hand, suggests the use of empirical data analysis, in particular empirical membership function, a CBF based approach, for property recommendation.

2.6.1 Love-Hate Square Counting

The square counting method, proposed by Kong et al. [2012] is a model based collaborative filtering where the aim is to predict if a user will like/dislike an item (unobserved link), and in order to achieve this, it relies on if other users have likes/disliked other items (observed link). For, e.g. if users u_1 and u_2 both like item i_1 (observed links) and user u_2 likes i_2 (observed links), then one can assume that user u_1 would like item i_2 as well (unobserved link). Similarly, if users u_1 and u_2 both dislike item i_1 (observed links) and user u_2 dislikes i_2 (observed links), then one can assume that user u_1 would dislike item i_2 as well.

Given users u_1 and u_2 and items i_1 and i_2, there are eight possible values of the configuration $u_1 - i_2 - u_2 - i_1$ as depicted in Figure 2.6(left), depending on whether the user u_1 likes/dislikes $(+/-)$ an item i_2 that is liked/disliked $(+/-)$ by a user u_2 who likes/dislikes $(+/-)$ the item i_1 encoded by 0-7. Kong et al. [2012] proposes predicting the value of the unobserved link $u_1 - i_1$ from all possible users $u_1 - i_2 - u_2 - i_1$ values, and implements this efficiently through a bipartite graph with user and item nodes with edge denoting like/dislike.

Figure 2.6(right) shows an example of such a bipartite graph. There are four users u_1, u_2, u_3 and u_4 and three items i_1, i_2 and i_3 besides the target user u_{tg} and target item i_{tg}. An eight dimensional feature vector $\mathbf{c}$ corresponding to the pair $u_{tg} - i_{tg}$ is constructed by counting all possible configurations $u_{tg} - i - u - i_{tg}$. For example, there is no link that resembles the configuration -,-,+ (configuration 1) or +,+,+ (configuration 7). Hence these configuration counts are 0 in the vector $\mathbf{c}$. Similarly, we can count 2 -,+,+ configurations (configuration 3) through $i_2 - u_4$ and $i_1 - u_2$ respectively, and therefore the corresponding value in $\mathbf{c}$ is 2. Kong et al. [2012] used observed links to train a classifier that converts $\mathbf{c}$ into a like or dislike.

This model, like any other collaborative filtering models, is dependent on user's feedbacks. Therefore, it can work well where implicit or explicit user ratings are available, and it will fail when the data is sparse with too many missing links.

2.6.2 Empirical Fuzzy Sets

Angelov and Gu [2017] hints at using *empirical membership function* for property recommendation. Rather than relying on *hard* decisions on the features of a property, the use of *fuzzy logic* allows *soft* decisions through the use of membership functions. For example, rather than looking for a 2 bedroom flat for 700£ or less, the user can look for a 'cheap

[3] `https://www.zillow.com`
[4] `https://www.trulia.com`

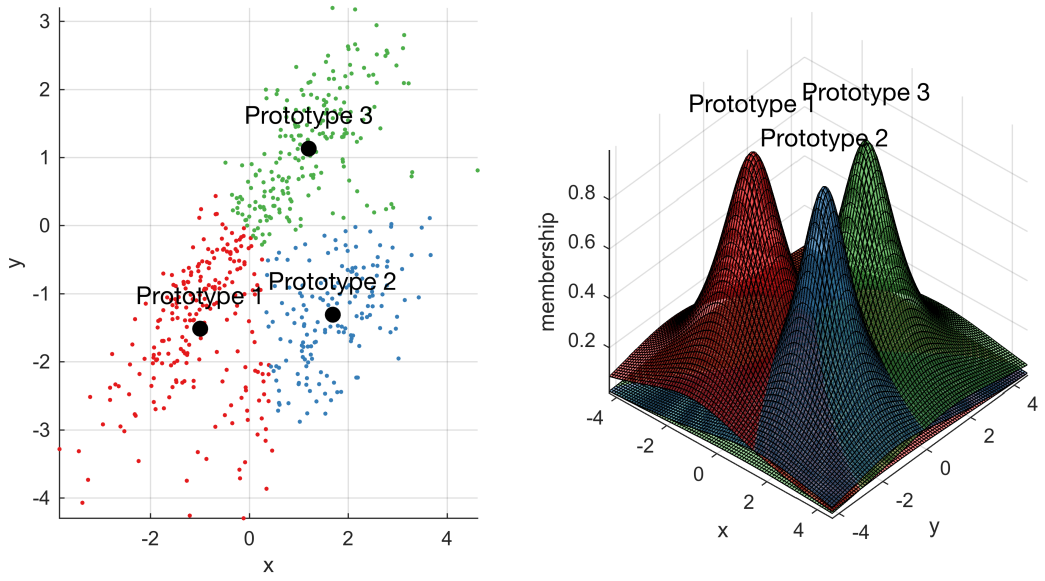


Figure 2.7: Construction of empirical membership functions.

medium sized' flat which assigns a possibility value over all possible available properties; a '2 bedroom flat for 700£ or less' should have more membership value than a '1 bedroom flat for 700£ or less' flat. Designing the membership function is, however, not a trivial task. Empirical membership function addresses this difficulty by estimating the membership functions from data without supervision. Figure 2.7 shows how empirical membership functions are constructed.

Given a set of *samples* X and a set of *prototypes* P, membership function is defined for each prototype $\mathbf{p}_k$[5]. This is done by first identifying its 'data cloud', i.e., set of samples $\mathbf{x}_i$ that are closest to the respective prototype $\mathbf{p}_k$ as

$$C_k = \{i : i \in \{1, \ldots, n\} \text{ and } \operatorname{argmin}_{k'} ||\mathbf{x}_i - \mathbf{p}_{k'}|| = k\}$$

Then, the membership function at a location $\mathbf{x}^*$ in expressed as

$$\text{eMF}_k(\mathbf{x}^*) = \frac{1}{1 + \frac{||\mathbf{x}^* - \mathbf{p}||^2}{\nu_k + ||\mu_k||^2}}$$

where

$$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} \mathbf{x}_i \text{ and } \nu_k = \frac{1}{|C_k|} \sum_{i \in C_k} ||\mathbf{x}_i||^2.$$

[5]Empirical membership function is defined through an arbitrary distance metric. We use Euclidean distances since our features are either continuous and discrete

Figure 2.8: Case base search and solution
source: snippet from the paper by Yuan et. al. on user oriented recommendation approach.

In the context of property recommendation one can use the properties the user have liked as prototypes, and find the corresponding membership functions. This implicitly captures the user's intention, e.g., if the user wants a 'cheap medium sized' flat. New properties then can be recommended to the user through its membership value [Angelov and Gu, 2017].

2.6.3 Case-Based Reasoning

In the general as well as in the traditional search engine reviewed in the section 1.1.1, search process of selecting attributes and checking results is an iterative process. Users check the search-results (if the result is not satisfactory) $\rightarrow$ go back to the search page again $\rightarrow$ change one of the selected attributes $\rightarrow$ again check the results. The process thus continues like a chain till the user has exhausted browsing or until the user finds satisfactory results. Thereby, in this process the search engine has already gone through multiple cases (searches), some similar, some different and found solutions for them. The scope to improve user-search-experience and reduce search time, the cases searched by the users in the past and the respective solutions can be stored and leveraged for new users.

[Yuan et al., 2013], proposes a user-oriented recommendation system for real estate websites via a combination of case-based reasoning (CBR) and an ontological structure [19]. CBR is a problem solving methodology that addresses a new problem by retrieving similar solved problems and reusing the results for solving current at hand. They used the customer requirements/constraints in a house search to construct an ontology framework. To begin with, a query is sent based on user's input. The query analyser then searches the database, and display related solution. Once the users narrow down their requirements, the

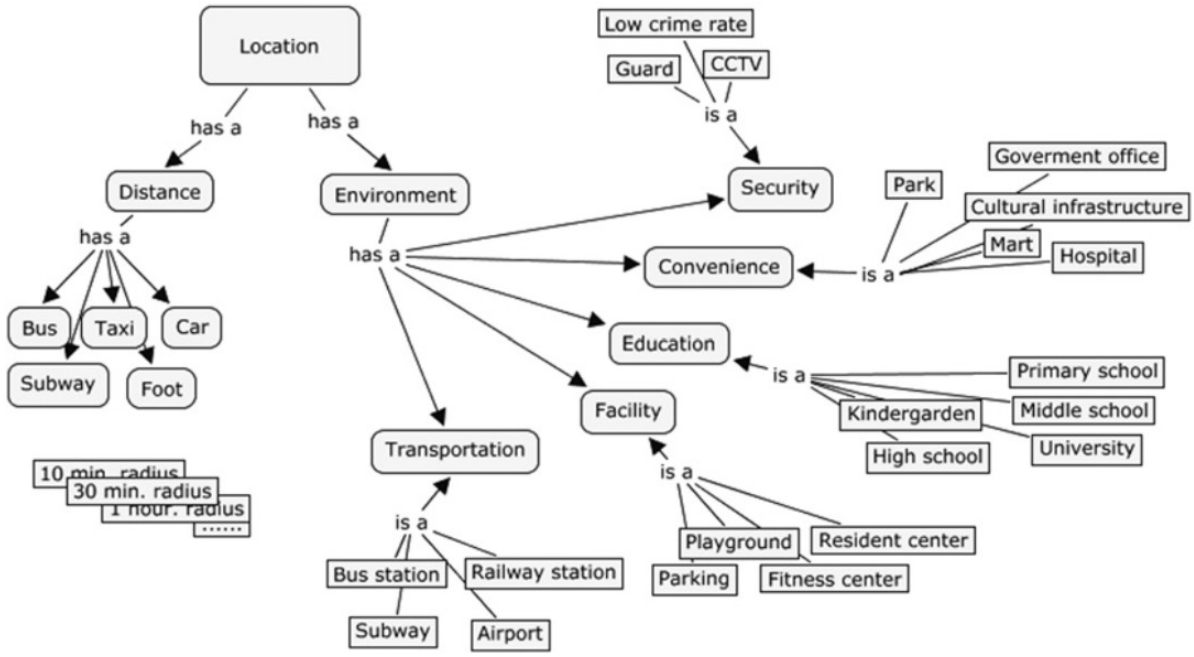


Figure 2.9: Shows the bottom down Ontology framework. Source: snippet from the paper by Yuan et. al.

sim- ilarity measurements compare the user's query and database cases in order to provide recommendations. The Figure 2.9 shows how the ontology framework gets narrowed down based on user preference and the Figure 2.8 shows how cases are built and stored to use it for recommendation when similar case is queried by the user again.

2.7 Evaluation for RSs

Evaluating the performance of a recommender system is a difficult problem but it has been studies extensively in the literature, see e.g., [Adomavicius and Tuzhilin, 2005b, Herlocker et al., 1999, Mooney and Roy, 2000a]. In §2.3, we have addressed the advantages and disadvantages of different recommendation system methods. All these attributes, e.g., accuracy, scalability, computational costs, play a major role in selecting the right approach for the application [Ricci et al., 2011]. [Herlocker et al., 2004], identified 3 different type of metrics to evaluate the accuracy of a recommender algorithm. The recommendation goal involves predicting ratings or classifying like, dislike. To compute the The ratings prediction accuracy which is usually a discreet value, mean absolute error (MAE) or root mean square (RMSE). For classification task of finding good items, precision and recall are used. The third metrics, rank accuracy, computes the ranking order accuracy for the application where ranking order of best to worst is important[Cacheda et al., 2011]. Metrics

of this type also measure the correlation between the predicted and observed classification. Other metrics can be coverage metrics which gives the percentage of items the model con correctly predict.

In the context of accuracy, a model can be evaluated in three different ways, first, with off-line evaluation where the performance of the recommender system is compared without user interaction, second, a small-scale survey can be conducted with a group of people by subjecting them to the system and gather their experience results, and third, a large scale on-line evaluation, where real user populations interact with the system [Ricci et al., 2011].

Since an online evaluation, although desired, is out of the scope of this thesis, we rely on offline evaluation. Two popular metrics for measuring the performance of a classification recommendation system, as discussed above are precision and recall, defined as [Cleverdon et al., 1968],

$$\text{precision} = \frac{|\{\text{relevant properties}\} \cap \{\text{retrieved properties}\}|}{|\{\text{retrieved properties}\}|}$$
$$\text{recall} = \frac{|\{\text{relevant properties}\} \cap \{\text{relevant properties}\}|}{|\{\text{relevant properties}\}|}.$$

In our case, $\{\text{relevant properties}\}$ are the properties listed in the test set $\mathcal{D}_u^{\text{tr}}$, and the $\{\text{retrieved properties}\}$ are the list of top N properties.

3 Methodologies

3.1 Data Preprocessing

In this section, we discuss the pre-processing steps for data cleaning, data transformation and dimensionality reduction. We have access to three datasets for i) property feature, ii) user profiles and iii) user-page logins.

3.1.1 Property features

The dataset contains information regarding **130,708 properties**. Each property is identified by a unique property-id, and is described by a combination of continuous (e.g., geographic location in terms of latitude and longitude), discrete (e.g., number of bedrooms and bathrooms) and categorical variables (e.g., is the property available for rent or for sale).

The information for each property can be recorded through two possible approaches, a) the user (owner of a property) can insert the details of the property manually via the company website, and b) property details are scraped automatically from other real estate websites. Scraping real estate websites is a continuous process to get new listings.

Beside the physical attributes of the property, additional ordinal variables are added to the property features, indicating proximity to local amenities. These variables rate the properties based on their proximity to park, station, school, etc. Each variables are rated between 0-3, where 0 stands for no proximity and 3 for high proximity. The scores are calculated automatically by a program written using Google map where the algorithm scores as follows: if (percentage $<$ 5) return 0 stars, if (percentage $<$ 50) return 1 star, if (percentage $<$ 83) return 2 stars, else return 3 stars. The Figure 3.1 shows the first 5 observations of the finalised property features file, where index is the property-Id.

Data Cleaning

Each property is represented by 45 variable. Variables like Id, Created date, Created by, Modified date, Address, Number, Zip, On appointment, Realtor Name, Phone, Website, Address.1, Number.1, Zip, Publishing status, Compares, Dislikes, Likes and Views are not required for recommendation, and were removed. Thus each property is represented by **20 features**. The 20 features are *nBeds, nBath, nGarage, rentOrSale, nToilet, price, landSqmt, propertySqmt, eduScore, greenScore, healthScore, meadowScore, parkScore, forestScore, shopScore, sportScore, trainScore, PTScore, lat and long.*[6] The 20-features were segregated

[6]The features are self explanatory.

	nBeds	nBath	nGarage	rentOrSale	nToilet	\
589f4424b68b4e1d209b4a51	2.0	1.0	1.0	0.0	1.0	
589f4424b68b4e1d209b4a52	2.0	1.0	1.0	1.0	0.0	
589f4426b68b4e1d209b4a53	2.0	1.0	0.0	1.0	0.0	
589f4426b68b4e1d209b4a54	2.0	1.0	1.0	0.0	0.0	
589f4426b68b4e1d209b4a55	2.0	1.0	0.0	1.0	0.0	

	price	landSqmt	propertySqmt	eduScore	\
589f4424b68b4e1d209b4a51	790.0	85.0	0.0	0.0	
589f4424b68b4e1d209b4a52	139000.0	0.0	75.0	0.0	
589f4426b68b4e1d209b4a53	149000.0	105.0	95.0	0.0	
589f4426b68b4e1d209b4a54	750.0	0.0	80.0	1.0	
589f4426b68b4e1d209b4a55	237900.0	0.0	90.0	0.0	

	greenScore	healthScore	meadowScore	parkScore	\
589f4424b68b4e1d209b4a51	0.0	0.0	0.0	0.0	
589f4424b68b4e1d209b4a52	0.0	0.0	0.0	0.0	
589f4426b68b4e1d209b4a53	0.0	0.0	0.0	0.0	
589f4426b68b4e1d209b4a54	0.0	0.0	0.0	0.0	
589f4426b68b4e1d209b4a55	0.0	0.0	0.0	0.0	

	shopScore	sportScore	trainScore	PTScore	\
589f4424b68b4e1d209b4a51	0.0	0.0	0.0	0.0	
589f4424b68b4e1d209b4a52	1.0	1.0	1.0	0.0	
589f4426b68b4e1d209b4a53	0.0	1.0	1.0	0.0	
589f4426b68b4e1d209b4a54	1.0	1.0	1.0	1.0	
589f4426b68b4e1d209b4a55	0.0	1.0	0.0	0.0	

	lat	long
589f4424b68b4e1d209b4a51	51.230801	4.412363
589f4424b68b4e1d209b4a52	51.255829	4.436802
589f4426b68b4e1d209b4a53	51.208923	4.471614
589f4426b68b4e1d209b4a54	51.226425	4.401914
589f4426b68b4e1d209b4a55	51.185062	5.103232

Figure 3.1: Property features data file with top 5 observations and 19 variables

into groups based on their data type, resulting in 14 continuous, 4 discreet (nBeds, nBath, nGarage, nToilet), and 1 categorical (RentOrSale) variable respectively. The rest of the features are continuous variables.

Missing values Since the project requires recommending properties to users and one of the key feature a user is interested in is the price of the property; any properties without price were removed from the dataset. Additionally, 66,465 property listings out of the total of 130,708 property listings did not have any environmental scores. However, these listings were retained based on the client specification to treat the missing values as 0, i.e., no proximity to related environmental variables. To ensure all the features are converted into respective data types, text data in the numerical columns were removed and vice-versa. Forest score was dropped from the data due to lack of data for any of the property reducing the features to 19.

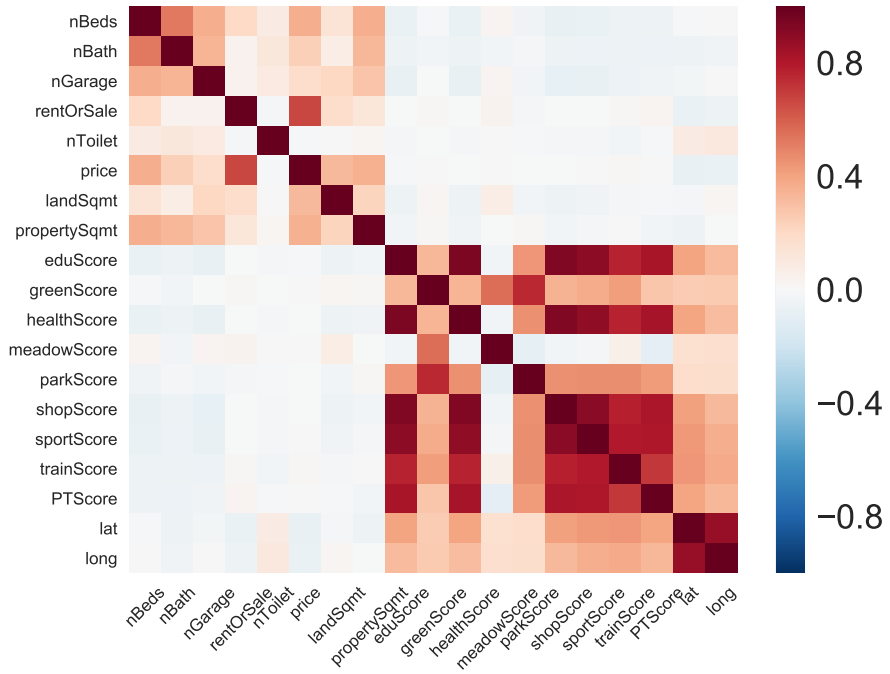
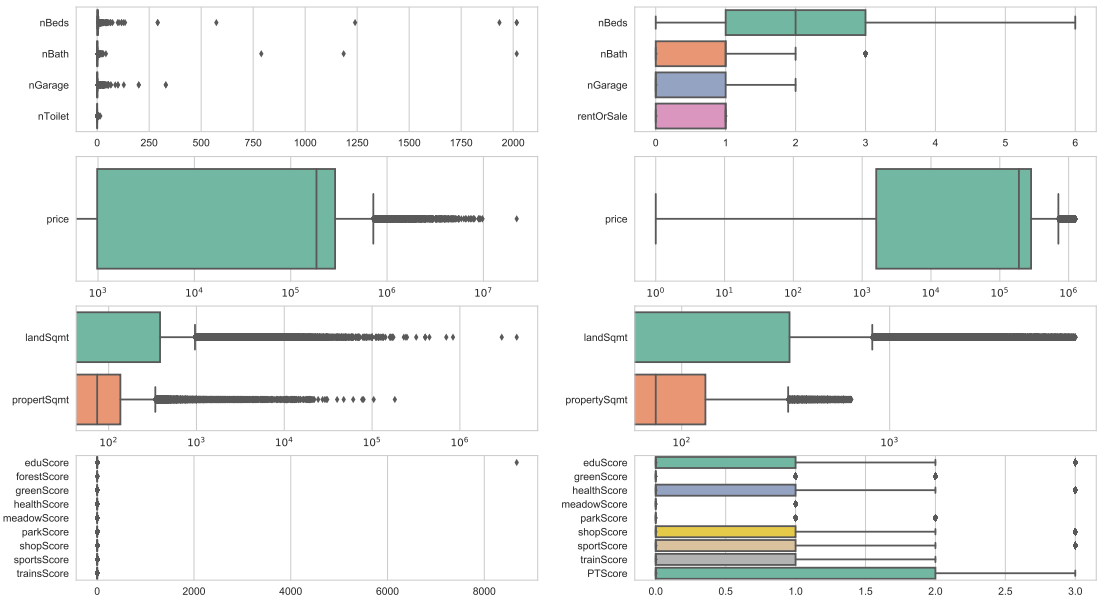


Figure 3.2: Shows correlations between 19 property features.

Bad values Figure 4.1 shows the distribution of all 19 features. We observe that some variables e.g., nBeds and nBath, show unrealistic values like 50 and 300 respectively. Such unrealistic values were identified are removed from the data. Additionally, we observe that some of the continuous variables are heavily right-skewed, and contains possible outliers. The presence of outliers can severely distort the outcome of an algorithm, e.g., if the algorithm requires estimating mean of the data. Therefore, we clean the data be removing the outliers. An outlier detection algorithm such as 'three sigma rule' assumes normal distribution of the data, and is not appropriate in this context. Other algorithm such as one based on inter-quartile range might also be inappropriate since it might get rid of low values that are not outlier. Therefore we removed bad values based on logic for example >10 beds, >3 baths were treated as bad values and removed.

Correlation Analysis We also checked the correlation between features. Figure 3.2 shows the correlation matrix. We can see minor box diagonal structure correlation close to environmental score, it is expected based on the latitude and longitude.

The resulting property features matrix is of dimension (115882, 19).



(a) Boxplot with bad-values **(b)** Boxplot without bad-values

Figure 3.3: Box-Plot before and after removing bad-values

Data transformation

We observe that the scales of the features varies drastically, e.g., the price ranges from 0 to 720,700 where as nBeds and nBath varies from 0 to 6. For methodologies based on the Euclidean distance measures, bringing all the features to one scale is imperative. One can either try different distance methods measures without standardizing the data or standardize the data and try Euclidean distance. We use z-score normalization[7] as follows

$$z = \frac{(x - \mu)}{\sigma}$$

where μ and σ are the mean and standard deviation of x.

3.1.2 Explicit and Implicit Data Collection

In order to make any recommendations, the system has to collect data. The ultimate goal of collection the data is to get an idea of user preferences, which can later be used to make predictions on future user preferences. There are two ways to gather the data. The first method is to 1 ask for explicit ratings from a user, typically on a concrete rating scale (such as rating a movie from one to five stars). The second is to gather data implicitly as the user

[7]One can also use the min-max scaling as normalization as follows $x_{norm} = (x - x_{min})/(x_{max} - x_{min})$. However, x_{max} can be sensitive to outliers, so we avoid this normalization.

is in the domain of the system - that is, to log the actions of a user on the site. Explicit data gathering is easy to work with. The ratings that a user provides can be directly interpreted as the user's preferences, making it easier to make extrapolations from data to predict future ratings. However, the drawback with explicit data is that it puts the responsibility of data collection on the user, who may not want to take time to enter ratings. On the other hand, implicit data is easy to collect in large quantities without any extra effort on the part of the user. Unfortunately, it is much more difficult to work with. The goal is to convert user behavior into user preferences, but it requires getting over one hurdle: how exactly does one infer preference based on actions in a system? This can be a difficult question to answer. Of course, these two methods of gathering data are not mutually exclusive. A combination of the two have the possibility for the best overall results - one could gain the advantages of explicit voting when the user chooses to rate items, and could still make recommendations when the user does not rate items by implicitly collecting data.

User profile: Explicit Data

The dataset contains information regarding **11,118** users. The data is extracted based on the users' input on the company website. It contains property features, e.g., nBeds, nBath, environment scores, and price [mean:4900000 and max:50,000,000] preferred by the users in addition to features like properties viewed, liked, compared and disliked by the users. We grouped the properties viewed, liked and compared by the user into one feature called 'liked'[8] A total of **107 users** were identified for active participation out of 11,118 users in providing the explicit information.

We did not utilise the user profile features from the user profile since the user search criteria and the properties user liked as shown in the Figure 3.5 did not match the actual preference of the user.

User Page login: Implicit Data

Although there are some explicit information available on which properties the users like, this is rather scarce. Therefore, we looked at capturing the likes and dislikes of a user implicitly from user page login information. The user page login data provides information on the time spent by each user on each website page. To extract property specific information, the time spent on the property pages were extracted. We classified the login time less than 21 secs (0.40 quantile) including 0 on a property as the user dislikes

[8]In future, when enough user data is available, these features can be used to rank the likeness instead of grouping them into binary indicator, like or dislike.

	Id	Likes	Dislikes	
0	58a0c2b9b68b5e0bc8a2d9fa	NaN	NaN	
1	58a0c658b68b600bc8e66e30	NaN	NaN	
2	58a0dab8b68b610bc88065e5	NaN	NaN	
3	58a0e8f5b68b660bc85772b6	NaN	NaN	
4	58a0c65ab68b600bc8e66e31	NaN	NaN	

	Compares	Views	Type	\
0	NaN	NaN	For sale	
1	NaN	NaN	For sale	
2	NaN	NaN	For sale	
3	NaN	NaN	For sale	
4	589f4459b68b4e1d209b4bc9, 589f4456b68b4e1d209b...	NaN	For sale	

	Categories	Bedrooms	Bathrooms	Garages	...	\
0	House, Appartement	NaN	NaN	NaN	...	
1	NaN	NaN	NaN	NaN	...	
2	NaN	0.0	2.0	NaN	...	
3	NaN	NaN	NaN	NaN	...	
4	House	NaN	NaN	NaN	...	

	Max. price	Education score	Sport score	Shops score	PT score	\
0	336126.0	0.0	0.0	0.0	0.0	
1	NaN	0.0	0.0	0.0	0.0	
2	500000.0	0.0	0.0	0.0	0.0	
3	NaN	0.0	0.0	0.0	0.0	
4	NaN	0.0	0.0	0.0	0.0	

	Health score	Train score	Green score	Parks score	Forests score
0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0

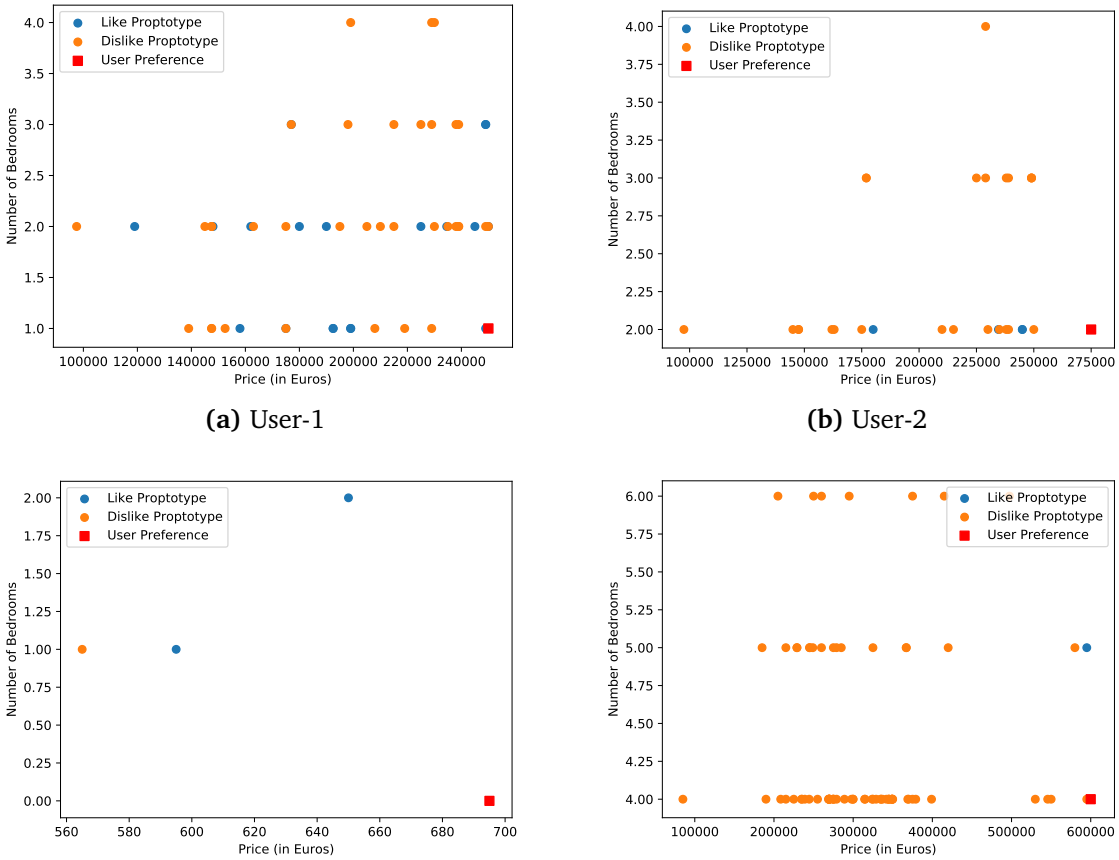
Figure 3.4: User profile data file with top 5 observations and 2 variables

property, login time more than 35 secs, were classified as likes property (0.60 quantile) and the in between as indecisive as shown in the user login time distribution/user in the Figure 3.6. However, the issue with this data was that the userId was not available to link it with the explicit data. We tried using IP-address however there was possibility of duplication with the users providing explicit data[9]. 1075 users were identified using implicit data in comparison to 107 using explicit, therefore implicit information plays an important role in learning user preference.

3.2 Collaborative Filtering

Collaborative filtering requires constructing the $n \times t$ user-item matrix Y where each entry $y_{i,u}$ denotes if the i-th entry is liked (1) or disliked (-1) by the u-th user. We denote the missing entries by NA. After removing items which have not been liked/disliked by any

[9]The client is instructed to assign userId to page-login data in future. The algorithm is built and can be used when userId is available.



(a) User-1 **(b)** User-2

(c) User-3 **(d)** User-4

Figure 3.5: Figure shows properties liked and disliked by the user and his/her preference based on the search conducted online

users, and users who have not liked/disliked any item we get a 155×647 matrix with sparsity as shown in the Figure 3.7.

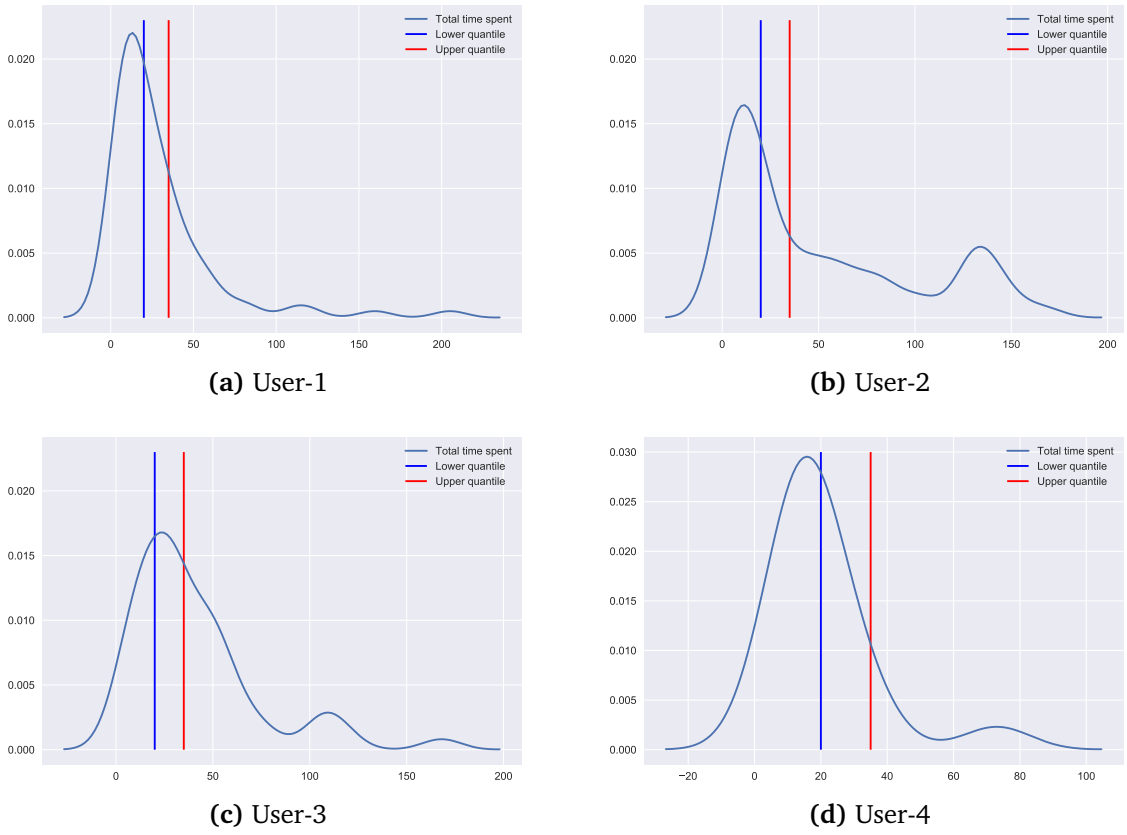
3.2.1 Neighbourhood Based

From section 2.2.2, for similarity based models, if the rating is binary, to compute similarity, we can use Jaccard similarity given as:

$$\text{sim}(u_1, u_2) = \frac{|\mathbf{y}_{u_1} \cap \mathbf{y}_{u_2}|}{|\mathbf{y}_{u_1} \cup \mathbf{y}_{u_2}|}$$

where $\mathbf{y}_u$ denotes the vector of all items with likes/dislikes by user u indicated.

However, Jaccard similarity ignores the ratings values and can have $\text{sim}(u_1, u_2) < \text{sim}(u_1, u_3)$ which goes against the intuition hence not suitable for the recommendation problem mentioned in Table 2.1. An alternate option is to use cosine similarity $\text{sim}(u_1, u_2) =$



(a) User-1 **(b)** User-2

(c) User-3 **(d)** User-4

Figure 3.6: Figure shows properties liked and disliked by the user and his/her preference based on the search conducted online

$\cos(\mathbf{y}_{u_1}, \mathbf{y}_{u_2})$. However, cosine similarity treats missing values in 1-5 ratings as 0, where 0 is a negative rating in 0-5 scale. Therefore, the magnitude of difference between $\text{sim}(u_1, u_2)$ and $\text{sim}(u_1, u_3)$ is not sound enough to show similarity and dissimilarity. This can be addressed by using centered cosine similarity where we first normalize the ratings by subtracting mean of the ratings for respective user ratings to center the ratings around 0 mean. Thereafter, we compute cosine similarity on the centered ratings $\mathbf{y}_u^c$ for user u. Centered cosine similarity is also known as Pearson Correlation, given as:

$$\text{sim}(u_1, u_2) = \cos(\mathbf{y}_{u_1}^c, \mathbf{y}_{u_2}^c).$$

Once the neighbourhood is identified, ratings $y_{u_1 i}$, for user u_1 and item i can be predicted by taking weighted average of the ratings given by the neighbourhood users based on the similarity values received by each user, as:

$$\hat{y}_{u_1 i} = \frac{\sum_{u_2 \in N(u_1)} \text{sim}(u_1, u_2)\ y_{u_2 i}}{\sum_{u_2 \in N(u_1)} \text{sim}(u_1, u_2)}$$

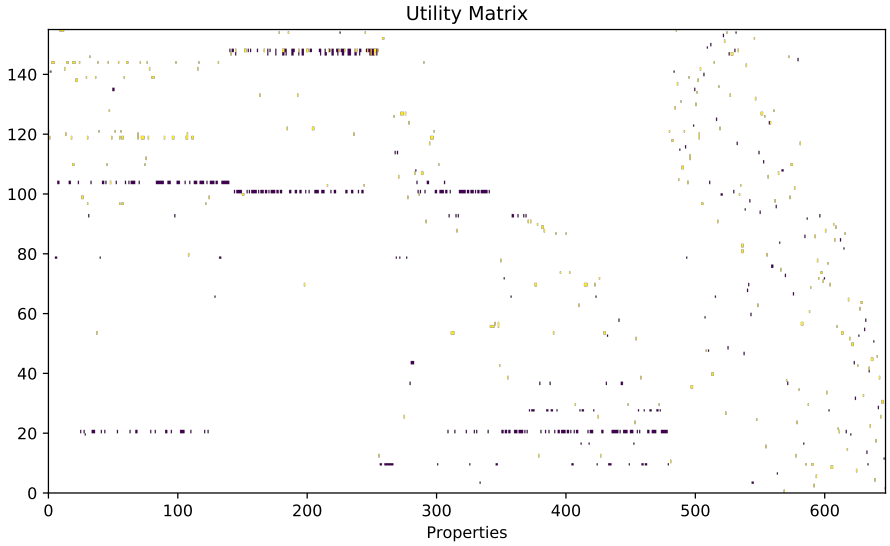


Figure 3.7: Sparse matrix of 155×647

where $N(u_1)$ is the set of k users similar to the user u_1 who also rated item i.

While user-user methods rely on the opinion of like-minded users to form the neighborhood, item-item approaches look similar items to form the neighborhood, i.e., an item-item approach models the preference of an user to an item based on the likes and dislikes to similar items by the same user. This method replicates the same similarity metrics and prediction functions as in user-user model as

$$\hat{y}_{ui_1} = \frac{\sum_{i_2 \epsilon N(i_1;u)} \text{sim}(i_1, i_2)\ y_{ui_2}}{\sum_{i_2 \epsilon N(i_1;u)} \text{sim}(i_1, i_2)}$$

where $N(i_1; u)$ are the set of items similar to item i_1, rated by the user j.

Model Implementation

In the neighbourhood based model for CF, we used Jaccard similarity to compute the similarity between users. We set the like/dislike value to 0,1 in the utility matrix of the dimension 155×647. This model is computationally intensive due to the interaction of a user with all the items and all the users in the matrix. The results were also poor due to sparsity of the utility matrix. GIven below is the example for top 5 similar users based on the Jaccard similarity, its very evident from the figures that the model failed to find similar users due to sparsity.

User Id - 58a0c2b9b68b5e0bc8a2d9fa

1. 0.000778
2. 0.000778
3. 0.000779
4. 0.000785
5. 0.000786
6. 0.000789

3.2.2 Model Based - Matrix Factorisation

Matrix factorization learns the latent matrices X and Θ by decomposing the utility matrix. We use nonnegative matrix factorization to decompose the utility matrix which assumes that both X and Θ are positive matrices. Since we are using nonnegative matrix factorization, we set the like/dislike values as 1 and 2. We set the latent dimension to 10. We observe that the training MSE 0.48 is quite poor potentially due to the sparsity of the matrix.

3.2.3 Model Based - Square Count

Square counting requires computing the interactions $u_1 - i_2 - u_2 - i_1$ in a bipartite graph (see §2.6.1). Since the utility matrix Y is rather small and quite sparse, we find these interactions by running a for loop over all u_1, u_2, i_1, i_2. This gives us 731 nonzero feature vector $\mathbf{c}$, out of which only 122 are unique. Furthermore, the classes are imbalanced (80-20). Figure 3.8 shows the (normalized) feature vectors we extract. We observe that the extracted feature and corresponding response variables enforces our intuition. For example, the configuration like-like-like ($l - l - l$) usually invokes positive response (like) while the configurations dislike-dislike-dislike ($d - d - d$) and dislike-like-like ($d - l - l$) invokes a negative response (dislike). Unfortunately, we observe that the sample size is not sufficient enough to learn a reasonable classifier (e.g., logistic regression).

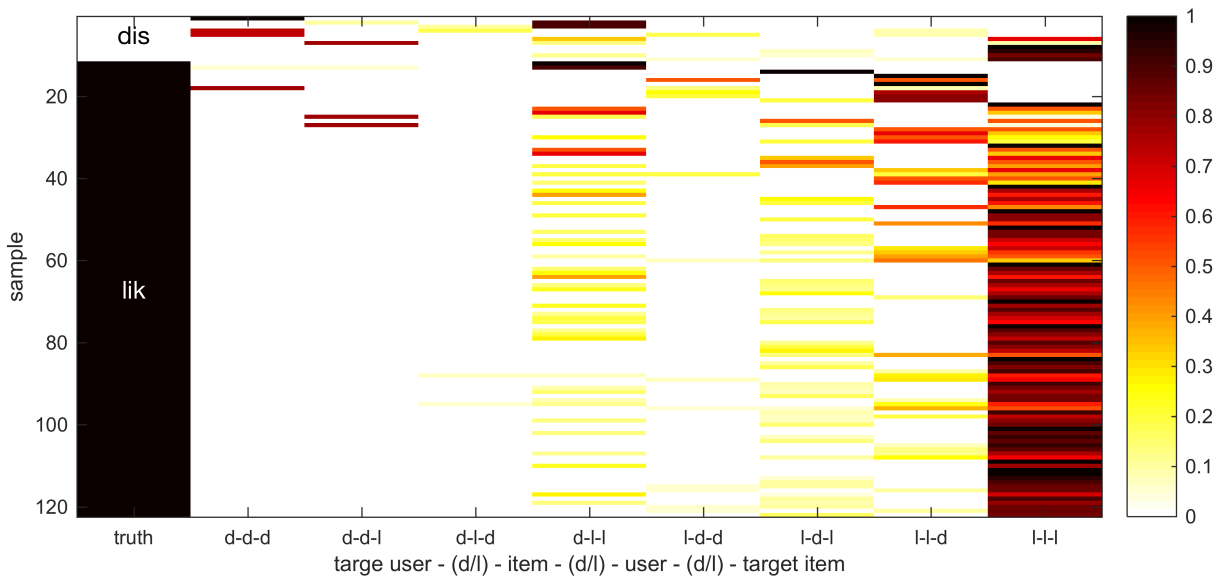


Figure 3.8: Feature vectors for square counting method.

3.3 Content Based Filtering

As we discussed in §2.2.1, CBF is a better approach for sparse data. We tried two baseline models, namely the logistic regression based approach and the cosine similarity based approach, and compared their performances against the proposed approach based on empirical data analysis. Its clear from the quality of the recommendation that the proposed approach outperforms the two baseline models, although the quality of recommendations from cosine-similarity is promising.

3.3.1 Baseline - Logistic Regression

The work below, is based on the tutorials provided by Andrew ng, on Recommender System on Coursera supported by Stanford University. In CBF, predicting the unknown ratings can be considered as a linear regression problem. The model expresses the rating y_{ui} as the inner product $\theta_u^\top \mathbf{x}_i$ where θ_u is the learning parameter for the u-th user, and $\mathbf{x}_i$ is the feature vector for the i-th item. To learn the parameter vector θ_u, the following optimization problem is solved.

$$\min_{\theta_u} \frac{1}{2} \sum_{i:r_{ui}=1} (\theta_u^\top \mathbf{x}_i - y_{ui})^2 + \frac{\lambda}{2}|\theta_u|$$

where $r_{ui} = 1$ if user u has rated item i else, $m_u = \sum_i r_{ui}$ is the number of items rated by user u, y_{ui} is the rating given by user u on item i, and λ is a regularization term.

Therefore given item features, $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and the ratings, we can estimate $\theta_1, \ldots, \theta_t$ to predict unknown ratings where n and t are the total number of items and users respectively. Table 3.1 illustrates the utility matrix with feature vector for each item.

Table 3.1: User-item pairwise ratings with features of each item

Items	user1 - $\theta^{(1)}$	user2 - $\theta^{(2)}$	user3 - $\theta^{(3)}$	user4 - $\theta^{(4)}$	Intercept - x_0	Feature1 - x_1	Feature2 - x_2
Item1 - $\mathbf{x}^1$	5	5	1	0	1	1.2	3.4
Item2 - $\mathbf{x}^2$	4	?	?	1	1	1.3	3.7
Item3 - $\mathbf{x}^3$	?	4	0	?	1	2	2.8
Item4 - $\mathbf{x}^4$	2	1	3	1	1	1.6	1.5
Item5 - $\mathbf{x}^5$	1	2	2	?	1	1.2	3.1

Model Implementation

We used logistic regression (LR) model as one of the baseline method for CBF. As we discuss in §4, we split the data into training and testing. However, since the highest number of properties liked or disliked is 35, it resulted in s small dataset affecting the classification. Therefore, the model does not perform well on the test data with an accuracy score of 0.36.

3.3.2 Baseline - Similarity Based CBF

The work is based on the lectures offered at Stanford on Recommendation Systems (Chapter 9). An alternate approach is to create the user profile as the (weighted) average of the items that the user has liked where the feature vectors are weighted by the ratings provided by the user, i.e.,

$$\theta_u = \frac{\sum_{i:r_{ui}=1} y_{ui}\mathbf{x}_i}{\sum_{i:r_{ui}=1} y_{ui}}.$$

Once the user profile is built the likeliness of an item to the user is evaluated by the cosine similarity $\cos(\phi_{ui})$ where

$$\cos(\phi_{ui}) = \frac{\theta_u^\top \mathbf{x}_i}{||\theta_u||||\mathbf{x}_i||},$$

and the items are ranked according to their similarities. This method, however, is computationally intensive as the similarity between each user and items require to be computed. The Table 3.2 shows how the user profile is extracted using profile of the items liked or disliked by the user

Table 3.2: User Profile and Item Profile

User Profile				Item Profile			
Users	Intercept - i_0	Feature1 - i_1	Feature2 - i_2	Items	Intercept - x_0	Feature1 - x_1	Feature2 - x_2
user1 - i^1	1	1.46	2.90	Item1 - x^1	1	1.20	3.40
user2 - i^2	1	1.53	2.78	Item2 - x^2	1	1.30	3.70
user3 - i^3	1	1.60	2.47	Item3 - x^3	1	2.00	2.80
user4 - i^4	1	1.40	2.30	Item4 - x^4	1	1.60	1.50
user5 - i^5	1	1.20	3.10	Item5 - x^5	1	1.20	3.10

Model Implementation

We used cosine-similarity metrics to evaluate the likeness of an item by the user. We took the weighted average of the items (in the training set) liked by the user to create a user profile, and then computed the cosine similarity between the user profile and the remaining properties. The minimum cosine-similarity value in the test dataset is 0.91. With this threshold in mind, we checked the similarity values for the properties in the top-10 recommendations. All the recommended top-10 prooperties showed, similarity values >0.91 which suggests that these properties have high likelihood of being liked by the user. We also compared the property in the top 10 recommendations with 10 properties in the test data set. We see 2/10 properties appear in the top-10 recommendations.

3.3.3 Proposed - Empirical Fuzzy Membership

Empirical MF Approach

To understand the computation of Empirical MF, let us take a look at how the traditional FR system works. In a traditional Mamdani type or Takagi-Sugeno type FRB system, to segregate the data into a group or clusters, the linguistic terms and the area of influence needs should be defined. For e.g., to build a cluster of low, medium and expensive house, Mamdani type or Takagi-Sugeno type FRB system requires the FRs expressed linguistically, as

$$\text{Rule}_1 : \textit{IF}\ (\text{price is between 100,000 and 250,000})\ \text{THEN}\ (\text{low-price})$$

Similar such FRs have to be defined for medium and expensive houses. Here the ***IF*** part of the rule is called the antecedent part and then-part the consequent part. However, in AnYa type FRB system [Angelov and Yager, 2012], instead, three data clouds are formed around the prototypes used in the aforementioned example. These data clouds are then

used as the IF part, simplifying the need to define linguistic terms and the areas of influence as with the traditional FRB. Thus for a dataset without a categorical variable, the antecedent part of empirical FRB on the AnYa type of FRB can be expressed as:

$$Rule_i : \textit{IF } (\boldsymbol{x} \sim \textbf{\textit{prototype}}_i) \textit{ THEN } (Class_i) \tag{3.1}$$

For a dataset that includes both categorical and continuous/discreet variables, $\mathbf{X} = [c^1, ..., c^m, \mathbf{x}]$, where c^j is the j^{th} categorical variable and $\mathbf{x}$ is the continuous/discreet variable of the set, empirical fuzzy rule takes the general form as:

$$\textit{IF } \left(c^1 = \Gamma_i^1\right) \textit{ AND...AND } (c^m = \Gamma_i^m)$$

$$Rule_i : \textit{THENIF } (\boldsymbol{x} \sim \textbf{\textit{prototype}}_i) \textit{ THEN } (Class_i) \tag{3.2}$$

where Γ_i^j, the j^{th} categorical variable of the i^{th} prototype can only take on one value from c^j, the j^{th} categorical variable. Thus the output of the categorical (IF) part in the proposed empirical FR is a Boolean ('true' or 'false' only) expressed as:

$$b_i^j\left(c^j\right) = \left\{ \begin{array}{ll} 1 & c^j = \Gamma_i^j \\ 0 & c^j \neq \Gamma_i^j \end{array} \right] \tag{3.3}$$

Therefore the degree of membership for $\mathbf{X}$ is defined as the product of the outputs of the categorical (IF) part and the continuous and/or discrete (THENIF) part as:

$$\lambda_i\left(\mathbf{X}\right) = \prod_{j=1}^{m} b_i^j\left(c^j\right).\mu_i\left(\mathbf{x}\right) \tag{3.4}$$

where $\mu_i\left(\mathbf{x}\right)$ is the empirical MF of the empirical FR for the continuous/discreet part which is computed automatically from the data cloud around the prototypes defined. For Euclidean type of distance, it takes a form of Cauchy function:

$$\mu_i\left(\mathbf{x}\right) = \frac{1}{1 + \dfrac{||\mathbf{x} - \mathbf{p}_i||^2}{X_i - ||\mathbf{p}_i||^2}} \tag{3.5}$$

where $\mathbf{p}_i$ is the i^{th} prototype of the data cloud and X_i is the average scalar product of the members of the data cloud.

The prototype can therefore attract similar items to it to form shape-free density clouds around the prototypes automatically Angelov and Gu [2017]. It also addresses the property

Algorithm 1 computeCluster ▷ Compute cluster assigments of X given P

Require: X ▷ $n \times m$ matrix of samples vs. features
Require: P ▷ $p \times m$ matrix of centers vs. features
1: **for** $i \in \{1, \ldots, n\}$ **do** ▷ Over all samples
2: **for** $j \in \{1, \ldots, p\}$ **do** ▷ Over all centers
3: $D[i, j] = ||X[i,] - P[j,]||$ ▷ Find distances, $||\mathbf{x}||^2 = \sum_t x^2[t]$
4: **end for**
5: $C[i] = \text{argmin}_{j'} D[i, j']$ ▷ Find index of minimum
6: **end for**
7: **return** C ▷ $n \times 1$ vector of $\{1, \ldots, p\}$ values

of close distance, average price which can differ from user to user. Once the user selects a property of his choice a data cloud of properties with degree of membership is formed around the cloud. The membership degree quantifies the grade of membership of the property to the fuzzy set. Based on the degrees of similarity of each house to the prototypes, recommendations are made to the user. The property with highest membership degree is chosen to make recommendations. The difference with the traditional FS and FRB is that the expert does not need to define the MF per variable; instead, possibly multimodal, density will be extracted automatically from the data and used as a MF in a vector form for all continuous variables. Since these are user specific clouds, the clouds learn from the multiple prototypes selected by the user to give personalised recommendation. This newly proposed way of fuzzy sets and FRB systems is suitable for big data. The Figure 2.7 below hows how the cloud is formed around the prototypes.

Empirical membership function based recommendation system requires requires computing the membership functions for each prototypes, i.e., properties liked by the user. As mentioned before, to compute the membership function, we use Euclidean distance on the standardised data. We use the same distance metric for computing the 'data clouds' related to each prototype, and assign each property to the closest prototype. Note that since we are doing this offline, we use the property data (without the training set) itself to compute the data cloud rather than estimating the related mean and norm statistics from a previously held-out dataset. The pseudo-code for computing the 'data clouds' and membership functions are described in Algorithm 1 and Algorithm 2. The Figure 3.9 shows the estimated membership function for a user over two variables, namely propertySqmt and price and the Figure 3.10 shows the estimated membership function for a user over two variables, namely shopScore and healthScore.

The idea of using membership function has been suggested by Angelov and Gu [2017]. The intuition behind this approach is that each property liked by the user defines a concept, and thus should be quantified through a membership function. The remaining properties

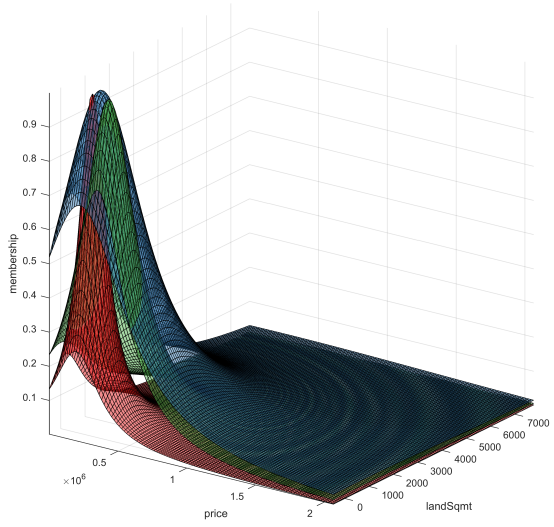


Figure 3.9: Empirical MF computed for property.sqmt (continuous) and price (continuous) for the 3 properties liked by the user. The peak of each data cloud is the prototype (properties liked)]

can then be sorted based on these membership values. We propose two different methods based on the empirical membership functions using only the liked properties of a user, and using both the liked and disliked properties of the user.

1. **Liked properties only** Here we sort the properties based on the maximum values over all prototypes. The intuition behind this approach is that if a property has high membership over *any* of the 'concept' suggested by the user, then it should be recommended to the user. This approach, however, has the drawback that it completely ignores the properties disliked by the user. We detail the pseudo-code of the method in Algorithm 3.

2. **Both liked and disliked properties** To resolve the issue raised in the previous method, we consider membership function for both liked and disliked prototypes. For each property, we find the maximum membership value over all liked prototypes, and over all disliked prototypes, and ignore all the properties for which the latter is higher. Then, we sort the remaining properties based on the maximum values over all liked prototypes. The intuition behind this approach is to differentiate concepts that the user likes and concepts that the user dislikes. For any property, if the concept 'dislike' carries higher significant than the concept 'like' then those properties should not be recommended to the user. The remaining properties can then be sorted based on their like values as usual. The drawback of this approach is that if the user likes and

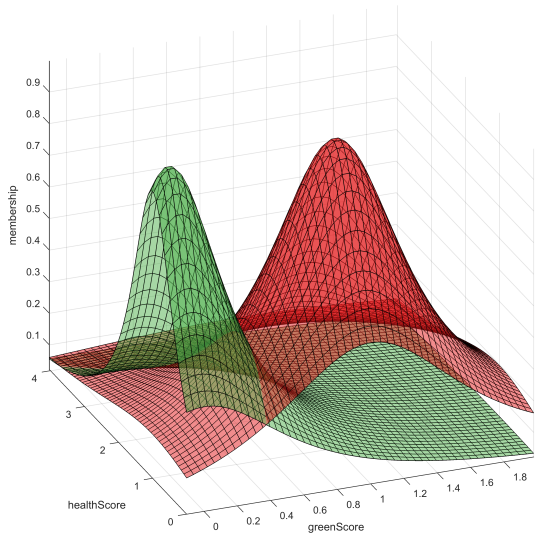


Figure 3.10: Empirical MF computed for greenScore (continuous) and healthScore (continuous) for the 2 properties liked by the user. The peak of each data cloud is the prototype (properties liked)]

dislikes properties that are close to each other then the algorithm might reject 'good' properties. We detail the pseudo-code of the method in Algorithm 4.

Algorithm 2 computeMembership ▷ Compute empirical membership functions at L

Require: X ▷ $n \times m$ matrix of samples vs. features
Require: P ▷ $p \times m$ matrix of prototypes vs. features
Require: L ▷ $l \times m$ matrix of locations vs. features
$C \leftarrow \mathsf{computeCluster}(X, P)$ ▷ $n \times 1$ vector of cluster assignments
for $j \in \{1, \ldots, p\}$ **do** ▷ Over all prototypes
  $C_i = \{s : s \in \{1, \ldots, n\} \text{ and } C[s] = j\}$ ▷ Get indices of samples in j-th cluster
  $\mu \leftarrow \frac{1}{|C_i|} \sum_{s \in C_i} X[s,]$ ▷ Mean of cluster
  $\nu \leftarrow \frac{1}{|C_i|} \sum_{s \in C_i} ||X[s,]||^2$ ▷ Norm of cluster
  for $i \in \{1, \ldots, l\}$ **do** ▷ Over all locations

$$\mathsf{eMF}[i,j] \leftarrow \frac{1}{1 + \frac{||L[j,] - P[i,]||^2}{\nu - ||\mu||^2}}$$

▷ Membership function

  end for
end for
return eMF ▷ $l \times p$ matrix of [0, 1] values

Algorithm 3 computeRecommendation ▷ Find top N recommendations

Require: propData ▷ $n \times m$ matrix of properties vs. features
Require: protData ▷ $p \times m$ matrix of prototypes vs. features
Require: N ▷ Number of recommendations
$\mathsf{eMF} \leftarrow \mathsf{computeMembership}(\mathsf{propData}, \mathsf{protData}, \mathsf{propData})$ ▷ $n \times p$ matrix of [0,1] values
for $i \in \{1, \ldots, n\}$ **do**
  $\mathsf{eMFmax}[i] \leftarrow \max_{p'} \mathsf{eMF}[i, p']$ ▷ Maximum value over all prototypes
end for ▷ $n \times 1$ vector of [0,1] values
return Indices of top N values in eMFmax

4 Evaluation

4.1 Experimental set-up

As discussed earlier, we have 115882 properties, and 15 users who have liked or disliked some of these properties. Our objective is to recommend properties to each of these users individually based on their respective likes and dislikes out of the remaining properties. Let $\mathcal{D}_u$ be the list of properties liked/disliked by the user u. We divide $\mathcal{D}_u$ in training $\mathcal{D}_u^{\text{tr}}$, and testing $\mathcal{D}_u^{\text{te}}$. We learn the model using $\mathcal{D}_u^{\text{tr}}$ properties, and recommend the top N properties from the remaining $115882 - |\mathcal{D}_u^{\text{te}}|$ properties. We observe if the properties listed in $\mathcal{D}_u^{\text{tr}}$ appears in the top N recommended properties.

Figure 4.1 shows the precision recall curves for different approaches for a particular user who liked/disliked 19/35 properties. We present results for several training-testing divisions

Algorithm 4 computeRecommendation2 ▷ Find top N recommendations

Require: propData ▷ $n \times m$ matrix of properties vs. features
Require: protDataLike ▷ $p_l \times m$ matrix of (liked) prototypes vs. features
Require: protDataDislike ▷ $p_d \times m$ matrix of (disliked) prototypes vs. features
Require: N ▷ Number of recommendations
eMFLikes $\leftarrow$ computeMembership(propData, protDataLike, propData)
▷ $n \times p_l$ matrix of [0,1] values
eMFDislikes $\leftarrow$ computeMembership(propData, protDataDislike, propData)
▷ $n \times p_d$ matrix of [0,1] values
for $i \in \{1, \ldots, n\}$ **do**
    if $\max_{p'}$ eMFLikes$[i, p'] > \max_{p'}$ eMFDislikes$[i, p']$ **then**
        eMFmax$[i] \leftarrow \max_{p'}$ eMFLikes$[i, p']$ ▷ Maximum value over liked prototypes
    else
        eMFmax$[i] \leftarrow 0$
    end if
end for ▷ $n \times 1$ vector of [0,1] values
return Indices of top N values in eMFmax

(25 random seeds), and observe that the results (e.g., area under the precision recall curve) varies significantly. This happens due to the lack of testing samples. However, nonetheless, we observe that best performance achieved by the proposed approach is better that the baseline approach. We present the number of true positives in top 25 recommendations averaged over all seeds in Table 4.1 which corroborates our observation from the figure.

Table 4.1: True postives at 25 recommendations with 10 likes in the test data

	min	**max**	**mean**
Logistic Regression	0	0	0
Cosine Similarity	0	1	0.08
eMF (Likes)	0	3	1.29
eMF (Likes Dislikes)	0	2	0.59

4.2 Qualitative assessment

We assess the quality of the recommended properties visually by plotting them on the space of property size and price for a particular user. For this user, we had 37 data points to train the model, and in a real estate based platform, we cannot expect users to like or dislike more than 20-30 properties to train a model since property search is not an ongoing e-commerce platform. Figure 4.2 shows the properties recommended by the logistic regression based

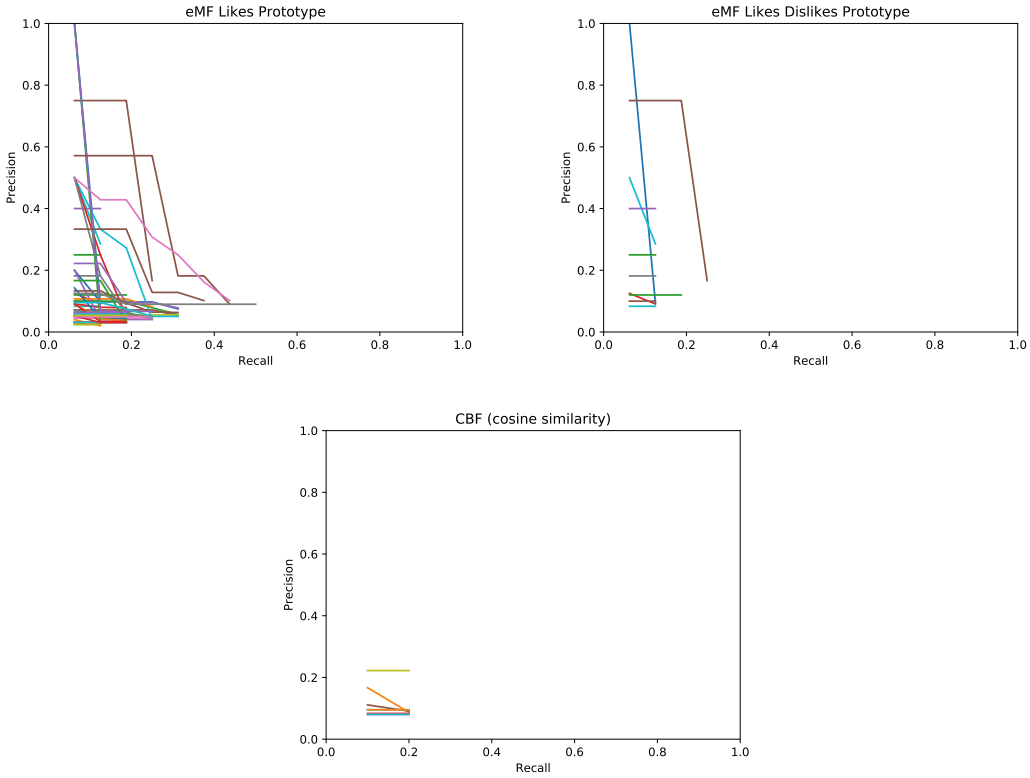


Figure 4.1: Precision Recall - Effect of small data sample and the randomness in sampling testing and training data. The above figure shows the precision-recall at different random seed sampling shown in different colours

approach. We observe that the recommended properties are far away from the observed prototypes indicating that the learning model failed to learn the likes and dislikes potentially due to lack of training samples.

Figure 4.3 shows that recommended properties based on the cosine similarity based method. We observe an improvement over the logistic regression based method in the sense that the recommended properties are close to the training prototypes. Since this method weights the user profile by both like and dislike prototypes, we do not observe any recommendation close to the dislike prototypes (e.g., in the cluster of samples at the bottom which is dominated by dislike prototypes). This shows the effectiveness of the model in recommending good quality recommendations.

Figure 4.4 shows the recommended properties using membership function based method with liked properties only. This method ignores disliked properties, and therefore, the recommendations included properties disliked by the user yet had high similarity with the properties liked by the user (e.g., the bottom cluster which is dominated by disliked properties). The quality of recommendations shows improvement over the recommendations

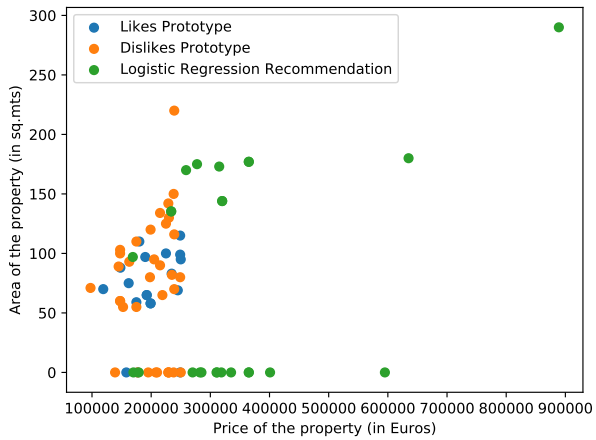


Figure 4.2: Recommendations based on Logistic Regression.

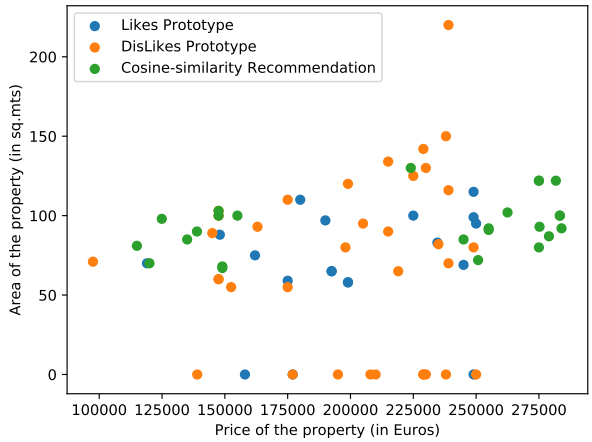


Figure 4.3: Recommendations based on Cosine Similarity computed for property.sqmt (continuous) and price (continuous) for the 4 properties liked and disliked by the user.

shown in the Figure 4.3 where the spread of the recommendations follow the prototypes, unlike the cosine similarity based recommendations where the recommendations look clustered in the extremes.

Figure 4.5 shows the recommended properties using membership function based method with both liked and disliked properties. This method uses both liked and disliked prototypes to recommend properties, similar to cosine similarity based method, and therefore we observe that it successfully removes the properties near the disliked properties from recommendation.

Recommending non-local properties We show that the property feature vector can be altered (which can also be seen as using a different distance metric with the same feature vector) to recommend property which is geographically separate from each other. This is

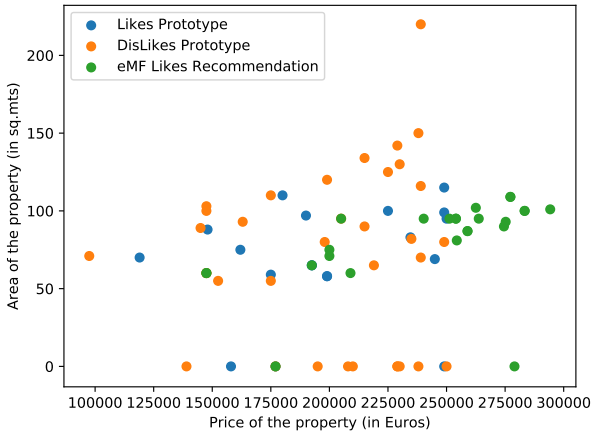


Figure 4.4: Recommendations based on Empirical MF computed for property.sqmt (continuous) and price (continuous) for the 4 properties liked by the user.

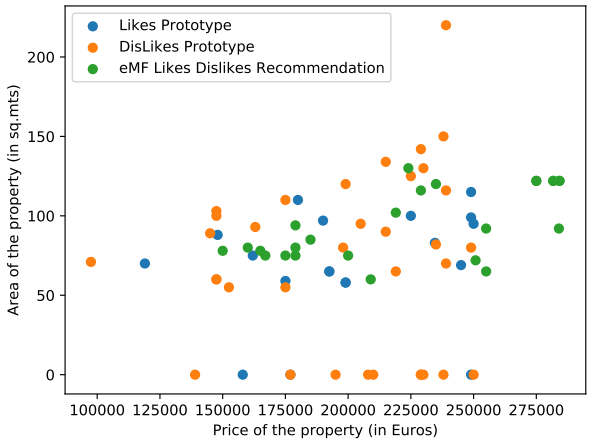


Figure 4.5: Recommendations based on Empirical MF computed for property.sqmt (continuous) and price (continuous) for the 4 properties liked and disliked by the user.

essential if the user is looking for similar properties but in a different neighborhood, or even a different city. To provide new location with the properties similar to the prototypes we use a 17 dimensional feature vector after removing the latitude and longitude features. Figure 4.6 shows the recommended properties using this method in a latitude versus longitude plot. We observe clusters of recommended properties in the plot which indicates properties from different cities. The spread of the cluster can be restricted by weighing the latitude and longitude differently rather than completely removing them from the feature vector.

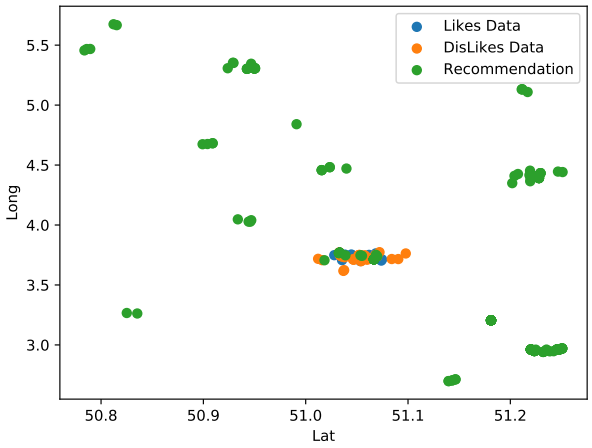


Figure 4.6: Recommendations based on training the eMF (LikeDislike-prototypes) algorithm ignoring the lat-long

5 Discussion

The findings from this research show that the proposed empirical membership function based recommender system approach is an effective model for personalisation in finding relevant properties based on the user's preference.

Building an algorithm for a recommender system is about computing similarities, either between users or between items or both. In this project, we reviewed and built four different state-of-the-art recommender systems from both collaborative filtering and content based filtering ideas. We also implemented two novel models, namely the square counting method, a collaborative filtering based approach, and proposed empirical membership function based approach, a content based filtering approach. The analysis of the collaborative filtering based approaches which use the user-item utility matrix demonstrates the negative influence of sparsity on the performance of the algorithm.

We used a memory-based and two model-based techniques from collaborative filtering where the experiments with memory-based techniques had to be terminated as the model could not evaluate similarities from a very sparse utility matrix. The model based approaches, i.e., matrix factorisation and square counting, on the other hand can be implemented successfully. Although the methods perform quite poorly, they show that utilising latent features to learn the low-dimensional vector space of users and items can improve prediction accuracy over memory-based model for very sparse matrix. However, with a dense matrix, memory based approaches outperform model based approaches, where model based techniques also use the steps similar to memory based techniques, like finding similar

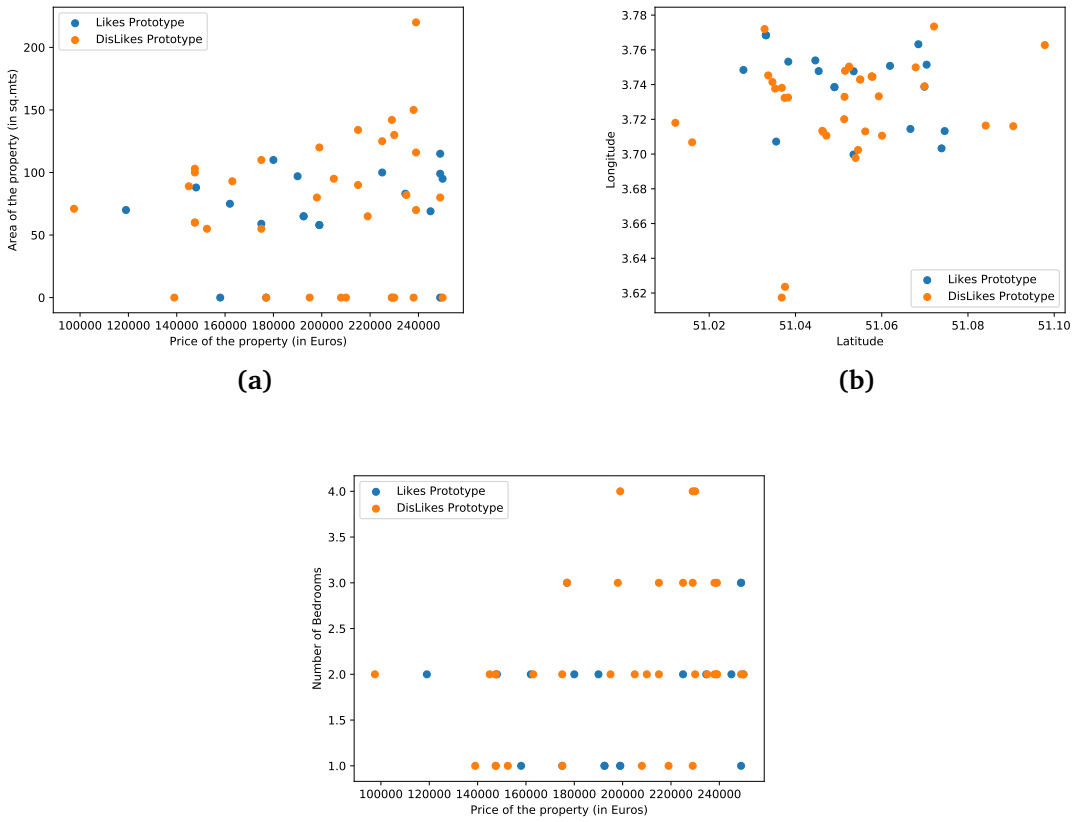


Figure 5.1: Properties liked and disliked by the same user plotted across different features

neighbourhood to make recommendations [Cacheda et al., 2011, Ozsoy, 2016].

The models based on content based filtering that exploit attributes (features) of an item to find similarities between items showed promising results in comparison to collaborative filtering. These models are not affected by the sparsity of the user rating data as they utilize features of the items to model the users. Content based filtering also enjoys the static nature of the similarity between items in comparison to the user based modes, so the similarities can be computed offline [Cacheda et al., 2011].

Within the content based filtering approaches, we used both parametric and non-parametric models. Parametric model like logistic regression showed extremely poor performance due to the lack of training samples since likelihood estimates requires a sufficiently large sample size to converge to their population value. We also used two non-parametric distance based model namely, the weighted cosine similarity based approach, and the empirical membership function based approach. Empirical membership based approach is built upon the inherent fuzzy nature of the property search where the user

might not want to put hard constraints on the property requirement but a similar property would be satisfactory.

The empirical membership for a property is calculated based on the properties liked by the user (prototypes), and therefore it retains more information in comparison to cosine similarity, where the distinct information from each prototypes is summarized into a user profile vector to compute the similarities between the user profile and the item attributes We expect this to be the potential reason of why the former perform better than the latter even though both are non-parametric similarity based models.

We also developed a novel method based on the empirical membership function using both prototypes that the user liked and the prototype the user disliked. While in the previous algorithm only the properties liked by the users were used to build the user-profiles, the proposed method is motivated from the cosine similarity based model which utilised the weighted average of both the properties liked and disliked by the user to create the user profiles. We observe that the proposed method perform more sensibly by not recommending properties near the dislikes prototypes.

Figure 5.1 shows the user rating behaviour for a user on three different variable pairs. We observe that the properties liked and disliked by the user overlap significantly in terms of their geographic location, area and price, and number of bedrooms, i.e., there are not specific feature based on which the two groups can be separated. This may explain the drop in the precision when both liked and disliked properties are used in empirical membership functions. Therefore, we believe that that there exist additional (latent) features beyond the recorded ones which play a crucial role in describing the property, and these should be studied in order to understand the user's choice better. Such properties can be, for example, the visual appeal such as color.

The property data and user data used in the experiment is from a young start up company. Technically the model was tried for only 3 users who liked and disliked >2 properties. The first element of both the sets given below represent the number of properties likes by a single user. After cleaning the data we could extract only 15 users who liked as well as disliked the properties.

Likes $\rightarrow 4, 19, 4, 2, 2, 2, 3, 1, 1, 1, 2, 8, 1, 1, 2$

Dislikes $\rightarrow 2, 35, 24, 1, 1, 52, 1, 1, 1, 1, 2, 1, 1, 84, 24$

Therefore all the methods that are sensitive to the sample size will not perform. Those models were evaluated under unfavourable conditions for the specific model. Having said that empirical MF have this advantage over other methods, that its not sensitive to sample size. Also empirical MF does not hold any assumptions. In future research all these methods must be evaluated under the favourable conditions. The table 5.1 shows steps in order, if

step 1 is not a fit, then next steps are not applicable. We see empirical fuzzy membership perform better.

Table 5.1: Method Fitness

		CF Methods			**CBF Methods**		
STEPS	METHODS	**Neighbourhood Based Similarity**	**Matrix Factorisation**	**Square Count**	**Logistic Regression**	**Cosine Similarity**	**Empirical Fuzzy Membership**
Step 1	Very Sparse	✗	✗	✗	✓	✓	✓
Step 2	Small Sample Size	NA	NA	NA	✗	✓	✓
Step 3	Scalability	NA	NA	NA	NA	✗	✓

6 Conclusions

The general aim of the thesis has been to develop a personalised recommender system for the real estate website Co-libry. The current recommendation system at Co-libry is knowledge based, and it recommends properties that match the user's criteria exactly. This is a constraint driven approach which does not model the user's personality. We suggest an alternate intent driven approach that looks at the properties liked and disliked by the user, and recommends properties that fit this user's personality.

A significant part of a project was devoted to data cleaning. We processed the raw data provided in 3 separate files, and stored it in a format that is suitable to work with in Python. We also performed extensive cleaning in terms of bad values, missing values, and outliers. Our exploratory analysis revealed interesting attributes of the data, e.g., although users inserted certain desired property attributes in knowledge based search engine, they liked properties which did not match these hard constraints. This corroborates the need for using an improved recommendation systems.

We started with a thorough literature study of existing work on recommendation systems. We found that the existing approaches are broadly classified in two categories, the collaborative filtering based approaches, and the content based filtering based approaches. Collaborative filtering based approach can be user-user based or item-item based, and both these approaches rely on the user-item utility matrix which indicates how a user has rated an item. We applied three related methods to our problem, and learned that this approach is not appropriate for our problem where the utility matrix is very sparse, i.e., we do not have sufficient users who have rated an item. This is a practical problem which may arise in the real estate since a user can only view a limited number of properties and he/she looks for property only a once or twice in his/her lifetime. This is a different situation, say for example, compared to music recommendation where a user listen to music on a daily basis and usually rates a significant amount of the music tracks either explicitly or implicitly.

Content based filtering approaches do not use the utility matrix but find similarities between items based on the item attributes. We learned that this is a more sensible approach to our problem. However, not all methods in this framework are suitable for us. For example, the regression based method requires a user to rate sufficient number of properties to build a user profile. Unfortunately, a user only rates very few properties. Therefore, we investigate other method that are free from this problem, and can function even if the user rates a single property. Along this line, we explored two methods which are in some sense equivalent to the idea of the 'nearest neighbors'. The first approach use the cosine similarity for finding similar properties, whereas the second approach uses

the empirical membership function as a similarity metric. While the former approach is computationally simpler, the latter performs better since it uses the rates properties directly rather then summarizing them through weighted mean. Through our experimental analysis, both quantitative and qualitative, we showed this to be true.

Although we successfully implemented the recommendation systems, we learned that the performance evaluation in this context can be problematic. Ideally, one can only test if a recommendation system is working is by showing the results to the user to see if the user agrees with the finding. This type of evaluation criteria also enables the design of an online system which updates its parameter based on the new likes and dislikes the user provides. However, this was out of the scope of this thesis. We evaluated our method offline by splitting the available ratings in training and testing. It would be interesting to explore these aspects of the system to better understand how the recommendation system fairs compared to the existing one.

Table A.1: Feature Description of PropertyList data and UserProfile data

SN	PropertyList	UserProfile	Variable Type	Variable Description
1	-	Dislikes	ResponseVar	Property Ids disliked by the user (propertyId can be linked to the properties in the propertyList)
2	Dislikes	-	Discrete	Number of users disliked this property
3	-	Likes	ResponseVar	Property Ids liked by the user
4	Likes	-	Discrete	Number of users liked this property
5	-	Views	ResponseVar	Property Ids viewed by the user
6	Views	-	Discrete	Number of users viewed this property
7	-	Compares	ResponseVar	Property Ids compared by the user
8	Compares	-	Continous	Number of users compared this property to other
9	Category	Categories	Categorical	10 different Property types - House Garage, Shop, etc
10	Sale Type	Type	Categorical	Sale/Rent
11	City	Locations	Categorical	City the property is located in
12	Lat	-	Continous	Latitude of the property
13	Lng	-	Continous	Longitude of the property
14	-	Commutes	Continous	Regular commute location given in Lat Long
15		CustomDrawn	Continous	Preferred area given in 4 Lats and 4 Longs
16	-	Min. price	Continous	Preferred min price of the property in euros
17	Price	-	Continous	Price of the property in euros
18	-	Max. price	Continous	Preferred max price of the property in euros
19	Landspace	-	Continous	Landspace of the property in sq.meters
20	Propertyspace	Min. surface	Continous	Property space of the property in sq.meters
21	-	Investment	Continous	Available amount to invest in euros
22	-	Income	Continous	Income of the user
23	-	Years	Continous	Preferred loan payment tenure
24	Bedrooms	Bedrooms	Discrete	Number of bedrooms
25	Bathrooms	Bathrooms	Discrete	Number of bathrooms
26	Garages	Garages	Discrete	Number of garages
27	Toilets	-	Discrete	Number of toilets
28	-	Kitchens	Discrete	Number of kitchens
29	Education score	Education score	Ordinal	1-5 Scores received / preferred by the user
30	Forest score	Forest score	Ordinal	1-5 Scores received / preferred by the user
31	Green score	Green score	Ordinal	1-5 Scores received / preferred by the user
32	Health score	Health score	Ordinal	1-5 Scores received / preferred by the user
33	Meadow score	Meadow score	Ordinal	1-5 Scores received / preferred by the user
34	Parks score	Parks score	Ordinal	1-5 Scores received / preferred by the user
35	Shops score	Shops score	Ordinal	1-5 Scores received / preferred by the user
36	Sports score	Sports score	Ordinal	1-5 Scores received / preferred by the user
37	Train score	Train score	Ordinal	1-5 Scores received / preferred by the user
38	PT score	PT score	Ordinal	1-5 Scores received / preferred by the user

A Summary of Variables

Table A.1 provides a summary of the variables available to us.

Table B.1: Summary table of raw-data and data files

FileName	Shape	Description
properties.csv	(130708, 47)	Raw data file with 130708 unique properties and their features [Not needed for analysis.]
profiles.csv	(11118, 34)	Raw data file with 11118 unique users and their preferences [Not needed for analysis]
analytics.csv	(61090, 10)	User Page visit information
p_nonStd_WO.csv	(115882, 19)	This file was created from properties.csv. It contains non- standardised property features data without missing values, bad values and outliers.
p_Std_WO.csv	(115882, 19)	This file was created from file 4. Contains standardised property features data without bad outliers
prototypedf.csv	(107, 3)	This file was created from the file profiles.csv. It contains 107 unique users and their preferred properties. Dislikes were excluded from this file, as these properties were treated as prototypes liked by the user.
dislikes.csv	(63, 2)	This file contains uId and the corresponding propertyIds disliked by the user.

B Summary of Data and Data Files

Table B.1 provides a summary of the datasets available to us.

Table C.1: Summary table of modular codes for eMF

Modular Code	Input	Output
prototypeCleanExtract (userData)	profiles.csv as numpy array	Numpy array of the list of propertiesIds for properties liked by each user. Call prototype[0] after running the function to get prototypes for user1.
extractPrototypeData (propertyData,prototypeList)	p_Std_WO and prototype[0] both as array	prot_data and prop_Data as numpy array and index of the property ID as list prot_data which is the prototype data for the respective user and prop_Data, is the property data filtered for sale or rent based on prototype
computeEuclDist (propertyData, prototypeData)	prot_data and prop_Data as numpy array extracted from **extractPrototypeData**	Numpy array of euclDist
computeCluster (propertyData, prototypeData)	prot_data and prop_Data as numpy array extracted from **extractPrototypeData**	minDistClust which is an array of clusters for each prototype
computeMembership (propertyData, prototypeData, location)	prot_data and prop_Data as numpy array extracted from extractPrototypeData and location if eMF needs to be calculated on different dataframe from property data	Numpy array of MF for each property
giveRecommendations (propertyData, prototypeData, location, recommendationSize, indices)	prot_data, prop_Data and index extracted from **extractPrototypeData** and location as numpy array if eMF needs to be calculated on different dataframe from property data. Specify recommendationsize manually	List of propertyIds of top recommendations, based on the recommendation size sepcified.

C Technical Implementation of Algorithm

The implementation of algorithm is done on Python. Modular codes and scripts were created for required functions. The input files are as mentioned in the Table B.1. Please refer to Table C.1 for the summary of modular codes and scripts available in Python to execute the above mentioned eMF algorithm.

References

Gediminas Adomavicius and Alexander Tuzhilin. Personalization technologies: a process-oriented perspective. *Communications of the ACM*, 48(10):83–90, 2005a.

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005b.

Nikhat Akhtar and Devendera Agarwal. A literature review of empirical studies of recommendation systems.

Plamen Angelov and Ronald Yager. A new type of simplified fuzzy rule-based system. *International Journal of General Systems*, 41(2):163–185, 2012.

Plamen Angelov, Xiaowei Gu, Dmitry Kangin, and Jose Principe. Empirical data analysis: a new tool for data analytics. In *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*, pages 000052–000059. IEEE, 2016.

Plamen P Angelov and Xiaowei Gu. Empirical fuzzy sets. *International Journal of Intelligent Systems*, 2017.

Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, Nov 2016. doi: 10.1007/s00799-015-0156-0.

Bettina Berendt, Andreas Hotho, and Gerd Stumme. Towards semantic web mining. *The Semantic Web—ISWC 2002*, pages 264–278, 2002.

Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1):2, 2011.

Cyril W Cleverdon, Jack Mills, and Michael Keen. Factors determining the performance of indexing systems. 1968.

Michael D Ekstrand, John T Riedl, Joseph A Konstan, et al. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173, 2011.

Usama M Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. *Advances in knowledge discovery and data mining*, volume 21. AAAI press Menlo Park, 1996.

David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992. doi: 10.1145/138859.138867.

Thomas Goodnight and Sandy Green. Rhetoric, risk, and markets: The dot-com bubble. *Quarterly Journal of Speech*, 96(2):115–140, 2010. doi: 10.1080/00335631003796669.

Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

Dietmar Jannach and Gerhard Friedrich. Tutorial: recommender systems. In *International Joint Conference on Artificial Intelligence Beijing*, 2013.

Jong-Hun Kim, Un-Gu Kang, and Jung-Hyun Lee. Content-based filtering for music recommendation based on ubiquitous computing. In *International Conference on Intelligent Information Processing*, pages 463–472. Springer, 2006.

Joseph S Kong, Kyle Teague, and Justin Kessler. The love-hate square counting method for recommender systems. In *Proceedings of KDD Cup 2011*, pages 249–261, 2012.

Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

Giles. C Lee, Kurt D. Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. pages 89–98, 1998. doi: 10.1145/276675.276685.

Greg Linden, Brent Smith, and Jeremy York. *Amazon.com recommendations: Itemtoitem collaborative filtering*. 2003.

Joel P Lucas, Nuno Luz, MaríA N Moreno, Ricardo Anacleto, Ana Almeida Figueiredo, and Constantino Martins. A hybrid recommendation approach for a tourism system. *Expert Systems with Applications*, 40(9):3532–3550, 2013.

Bradley N Miller, John T Ried, and Joseph A Konstan. Grouplens for usenet: Experiences in applying collaborative filtering to a social information system. In *From Usenet to CoWebs*, pages 206–231. Springer, 2003.

Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 195–204, New York, NY, USA, 2000a. ACM. ISBN 1-58113-231-X. doi: 10.1145/336597.336662.

Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204. ACM, 2000b.

Makbule Gulcin Ozsoy. From word embeddings to item recommendation. *arXiv preprint arXiv:1601.01356*, 2016.

Manos Papagelis, Ioannis Rousidis, Dimitris Plexousakis, and Elias Theoharopoulos. Incremental collaborative filtering for highly-scalable recommendation algorithms. *Foundations of Intelligent Systems*, pages 7–17, 2005.

Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3): 56–58, 1997.

F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. *Recommender Systems Handbook. Springer, Boston, MA*, 2011. doi: 10.1007/978-0-387-85820-3_1.

Kai Yu, Xiaowei Xu, Martin Ester, and Hans-Peter Kriegel. Selecting relevant instances for efficient and accurate collaborative filtering. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 239–246. ACM, 2001.

Xiaofang Yuan, Ji-Hyun Lee, Sun-Joong Kim, and Yoon-Hyun Kim. Toward a user-oriented recommendation system for real estate websites. *Information Systems*, 38(2):231–243, 2013.

Chun Zeng, Chun-Xiao Xing, and Li-Zhu Zhou. Similarity measure and instance selection for collaborative filtering. In *Proceedings of the 12th international conference on World Wide Web*, pages 652–658. ACM, 2003.

Yanchun Zhang, Guandong Xu, and Xiaofang Zhou. A latent usage approach for clustering web transaction and building user profile. In *ADMA*, pages 31–42. Springer, 2005.

Project Specification
Guide2Property

Snehal Nair

October 23, 2017

1 Introduction

Most of the online retailers understand the importance of this discovery and have addressed it by having in-built recommendation algorithm which can learn customer-profile and behaviour to identify items similar to the customers' liking for recommendation.

Guide2Property is a user-centric state of the art real estate one-stop shop. Based on a user profile combined with open datasets and AI, they provide a personal guidance and push-based content, contacts and offers. In this project we will research the effectiveness of personalisation for customers who are looking for places to lease/rent or buy. Some of the personalisation can be based on, user behaviour, similarities between properties and similarities between users.

The remainder of this paper is organised as follows. Section 2 discusses the problems, limitations, scope and research objective related to personalisation of the website, section 3 provides a brief overview of the personalisation process. Data collection process and a detailed description of data is discussed in section 4. The proposed approach is described in section 5. In the last section the project time-lines is provided.

2 Current Approach

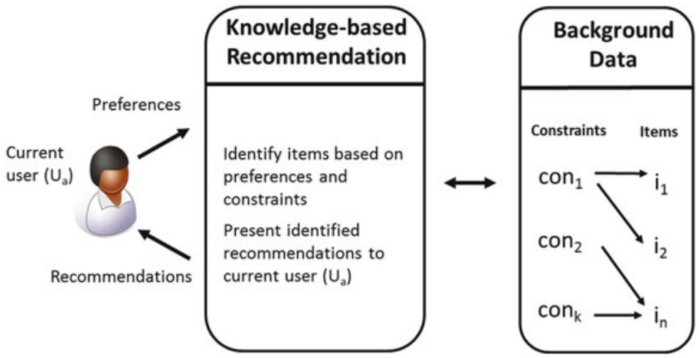
2.1 Overview of Current Recommender System

At Guide2Property, the traditional search robot is used as a recommendation engine. A traditional search robot builds queries based on the user input on the website and executes it in the database to bring out the search results to the user. On the website, the user first enters the input for e.g. properties for sale or properties for rent and this creates the first base query. Based on the next input given for e.g. city name, another query is created. The query is thus updated for every user input and the final query is executed on the property database to provide recommendation based on the search results. Thus, if a user enters the information like property on sale, 3 bedroom, in the city of Ghent. The respective query is created and then the traditional search robot will figure out which properties should be listed for a user request for 3-bedroom house in Ghent.

While this approach is fairly simple, there are few parameters that makes the search robot, powerful. For example, a user can enter a "Point of Interest" on the website. The user can then specify how far he wants to live from his point of interest, say, within 20 minutes distance by train. The query thus updates automatically to include this location-based information. Another striking feature is, a user can also enter an 'environment

score', which is a rating from 1-3 to denote how important these factors, e.g. health, green, shops, train, etc are to the user. Based on the open Google Maps data, the query updates to include these parameters in the property search. This system works well to provide recommendations based on the user's requests. A registered user can also save the requests and search it again on the website, each time the user re-visits it.

The current system follows a recommendation approach called Knowledge-based recommendation (see Fig.2) which relies on the following background data: (a) a set of rules (constraints) or similarity metrics and (b) a set of items. Depending on the given user requirements, rules (constraints) describe which items have to be recommended.



[H]

Figure 1: Knowledge-based recommendation (KBR) dataflow: users are entering their preferences and receive recommendations based on the interpretation of a set of rules (constraints) [3].

In the next section we will review the problems with the current system and the areas of improvement.

2.2 Current Architecture

This section briefly describes the architecture in the current system which includes database, scalability and centralized data store. The company is currently using two database platforms like SQL and MongoDB where smaller datasets like all registered users , request for contacting a seller are stored in the SQL database and big datasets like properties, user-search profiles, location-based data and in-house analytics (like user-clicks) and feeds from scraping other websites for new-listings is stored in MongoDB. Both the databases, including webserver are running on Microsoft Azure.

3 Problem Statement

In the current recommendation system, the search is based on geographic location, price, property space, financial factors and other information on surrounding area (health, shopping, school, parks, etc.) and other miscellaneous factors like mortgage calculator. It is purely knowledge-based recommendation coupled with location-based search optimization. The recommendation of a property therefore solely depends on the users understanding of his/her needs and experience with the property purchase/lease and depending on the given user requirements, rules (constraints) describes which items have to be recommended. An article from, "The race to create a 'smart' Google" by CNN Money, "Discovery is when something wonderful that you didn't know

existed, or didn't know how to ask for, finds you".

Thus one of the problems identified with the current recommender system is that, it fails to surprise the user with such "wonderful discoveries". Instead, only recommends the "obvious" properties, the type of properties a user searched for. This has been studied extensively in Exploitation v.s. Exploration strategy in reinforcement learning where exploitation makes the best decision given current information and exploration gathers more information.

How can we ensure explorations coupled with exploitation for the users? We can look at personalizing the recommendation for every individual or micro-segments of users. Personalisation is about building customer loyalty by building a meaningful one-to-one relationship through understanding the needs of each individual, and helping satisfy a goal that efficiently and knowledgeably addresses each individual's need in a given context [3]. In the sub-sections below I have tried to summarize some of the problems and areas of improvement.

3.1 Need for User-Profiling based on Personas

From the above section it is clear that one of the key requirement for personalization is, understanding the individual, micro-segmenting them by demographics and lifestyle is one way, to help system identify different personas in the segments. "Personas are archetypes built to identify our real users profile, needs, wants and expectations in order to design best possible experience for them". Here is an example of personas :

- A working couple of age 40-45 with two children, one of the children is school-age. And the couple loves staying away from the city buzz yet at a location where there are schools, hospitals and park close-by.
- A working couple of age 25-30 with two children, one of the children is school-age. Therefore the couple would like to have the convenience of school, hospital and parks close-by yet close to 'high-life' urban lifestyle, like fashion and entertainment.

Therefore for the goal to personalise, the system should be able to gather information to study personas of the users for the best possible user-profiling. One of the proposed recommendation approach once the user-profiling is carried out is collaborative filtering where you recommend properties to a new user based on the properties preferred by a user of similar persona.

3.2 Improve User-Search-Experience based on their Online-Search-Behaviour

In the general as well as in the current traditional search robot recommender system, search process of selecting attributes and checking results is an iterative process. Users check the search-results (if the result is not satisfactory) -> go back to the search page again -> change one of the selected attributes -> again check the results. The process thus continues like a chain till the user is exhausted browsing or until the user finds satisfactory results. Thereby, in this process the search robot has already gone through multiple cases (searches), some similar, some different and found solutions for them.

We have the scope to improve user-search-experience and reduce search time, if we can store the cases searched by the users in the past and the respective solutions and leverage them for new users. One of

the proposed recommendation approach based on this is case-based reasoning (CBR) where the system will have a case base of previously solved problems and their solutions. New problems are solved by transferring and adapting solutions that were used for similar problems in the past.

3.3 Improve Navigational-Convenience by clustering similar properties

Using the property and surroundings information we can build property clusters for similar properties. For e.g. one of the largest real-estate websites like Trulia provides property clusters based on the proximity to school, luxury homes, crime-rates etc. We can also leverage property description provided on each property for additional insights on the property. We can extract these additional property features by using NLP (Natural Language Processing) on the property description provided on each property.

Clustering the properties into major groups like proximity to school, new listings, water-side houses, etc, can improve navigational convenience. We can thereafter based on user-segment interest recommend properties from these clusters. A recommended approach to this is content-based recommendation where it recommends items based on a comparison between the content of the items and a user profile.

3.4 Future innovations

- **Build Visual-Value-Proposition by analysing Property-Images liked by the user** : Most of the preference listing about the property doesn't include the visual aspect. For example if we can identify from the image the type of room with attributes like for e.g. wooden flooring, high ceiling, lot of sunlight, etc then we can find and recommend similar property images to the user.
  The recommendation engine can bank on the users willingness to trade-off convenience and other preferences for their dream property. It can be achieved by extracting property features from the property images a user liked and recommending a similar property.

- **Build community-based personalisation by leveraging individual Facebook profile** : Community-based recommendation system recommends items based on the preferences of the users friends. This technique follows the epigram "Tell me who your friends are, and I will tell you who you are". [11, 12]. Evidence suggests that people tend to rely more on recommendations from their friends than on recommendations from similar but anonymous individuals [14].

4 Research Aims

With this research project, the client is looking at ways to build user loyalty, acquire more users and improve conversion rate by personalising it for it's users. Personalisation is about building customer loyalty by building a meaningful one-to-one relationship through understanding the needs of each individual, and helping satisfy a goal that efficiently and knowledgeably addresses each individual's need in a given context [3].

As strongly advocated in the popular press (Peppers and Rogers [157, 158]), it is not clear that targeting personalised offerings to individual consumers will always be better than for segments of consumers because of the trade-off between sparsity of data for individual consumers and heterogeneity of consumers within segments.

One of the client aims also included personalisation based on Facebook friends profile. Due to lack of data, personalisation based on Facebook data will be beyond the scope of this project and the stipulated time will depend on the marketing activities that can generate traffic and more loyal users who are willing to share primary Facebook details. The expected time is approximately 12-18 months.

Based on the client requirements and limitations discussed, the scope and research objective of this project includes building an improvised recommendation engine that can maximize the predictive accuracy of a recommendation to personalise the individual recommendations or build recommendation for micro-segments that includes both registered users and others.

5 Data

5.1 Data collection

Data is collected both explicitly and implicitly. The data is extracted through three different mediums, website which contains user-preference data based on explicit data provided by the users, the second medium is google analytics that provides implicit information on the time spent by each user on different web-pages and the last medium is web-scraping. The real-estate websites are scraped to collect newly available properties, this file also contains properties entered by users to sell or rent. There is scope to gather more consumer information through Google Analytics. It provides consumer information like:

- Demographics : Age and gender
- Interests : Affinity categories which includes wide range of interests Google tries to pin onto visitors and In-market categories which includes interests closely linked to the business.
- Location : Language (Based on Browser language or google account information) and Location (based on IP location) Behavior : News vs. returning (visitors)

5.2 Data description

User Profile : This data is collected from co-libry website based on the explicit information provided by the users. The explicit information mainly includes property preference. The key features required for personalisation is described below:

- Categorical variables : type of property, location (preferred cities given)and sales type (sales/rent)
- Continuous variables : commutes (place of daily commute given in latitude and longitude), custom drawn locations (given by latitude and longitude for four points to cover the area preferred), min-max price of the property, investment available, income of the user and loan-payment years preferred
- Discrete variables : number of bedrooms, bathrooms, garages, and kitchens
- Ordinal Variables : 1 to 3 scale (bad to good) property scoring for education, forest, green, health, meadows, parks, shops, sports, train, PT.

Property List : This data is extracted via two channels a) the user can input a property via the website b) scrape other real estate websites to get their properties. The data mainly consists of property related informations like :

- Categorical variables : type of property, location (preferred cities given) and sales type (sales/rent)

- Continuous variables : compares, latitude, longitude, price, landspace, and property space
- Discrete variables : number of dislikes, likes, views on property and number of bedrooms, bathrooms, garages, and kitchens
- Ordinal Variables : 1 to 3 scale (bad to good) property scoring for education, forest, green, health, meadows, parks, shops, sports, train, PT.

Table 1: Feature Description of PropertyList data and UserProfile data

SN	PropertyList	UserProfile	Variable Type	Variable Description
1	-	Dislikes	ResponseVar	Property Ids disliked by the user (propertyId can be linked to the properties in the propertyList)
2	Dislikes	-	Discrete	Number of users disliked this property
3	-	Likes	ResponseVar	Property Ids liked by the user
4	Likes	-	Discrete	Number of users liked this property
5	-	Views	ResponseVar	Property Ids viewed by the user
6	Views	-	Discrete	Number of users viewed this property
7	-	Compares	ResponseVar	Property Ids compared by the user
8	Compares	-	Continous	Number of users compared this property to other
9	Category	Categories	Categorical	10 different Property types - House Garage, Shop, etc
10	Sale Type	Type	Categorical	Sale/Rent
11	City	Locations	Categorical	City the property is located in
12	Lat	-	Continous	Latitude of the property
13	Lng	-	Continous	Longitude of the property
14	-	Commutes	Continous	Regular commute location given in Lat Long
15		CustomDrawn	Continous	Preferred area given in 4 Lats and 4 Longs
16	-	Min. price	Continous	Preferred min price of the property in euros
17	Price	-	Continous	Price of the property in euros
18	-	Max. price	Continous	Preferred max price of the property in euros
19	Landspace	-	Continous	Landspace of the property in sq.meters
20	Propertyspace	Min. surface	Continous	Property space of the property in sq.meters
21	-	Investment	Continous	Available amount to invest in euros
22	-	Income	Continous	Income of the user
23	-	Years	Continous	Preferred loan payment tenure
24	Bedrooms	Bedrooms	Discrete	Number of bedrooms
25	Bathrooms	Bathrooms	Discrete	Number of bathrooms
26	Garages	Garages	Discrete	Number of garages
27	Toilets	-	Discrete	Number of toilets
28	-	Kitchens	Discrete	Number of kitchens
29	Education score	Education score	Ordinal	1-3 Scores received / preferred by the user
30	Forest score	Forest score	Ordinal	1-3 Scores received / preferred by the user
31	Green score	Green score	Ordinal	1-3 Scores received / preferred by the user
32	Health score	Health score	Ordinal	1-3 Scores received / preferred by the user
33	Meadow score	Meadow score	Ordinal	1-3 Scores received / preferred by the user
34	Parks score	Parks score	Ordinal	1-3 Scores received / preferred by the user
35	Shops score	Shops score	Ordinal	1-3 Scores received / preferred by the user
36	Sports score	Sports score	Ordinal	1-3 Scores received / preferred by the user
37	Train score	Train score	Ordinal	1-3 Scores received / preferred by the user
38	PT score	PT score	Ordinal	1-3 Scores received / preferred by the user

6 Methodology

6.1 Approaches followed by Zillow and Trulia

Zillow, Trulia are topmost real estate websites in the world. The below figure gives a snapshot of the architecture at Zillow. Trulia followed the same architecture as it was acquired by Zillow.

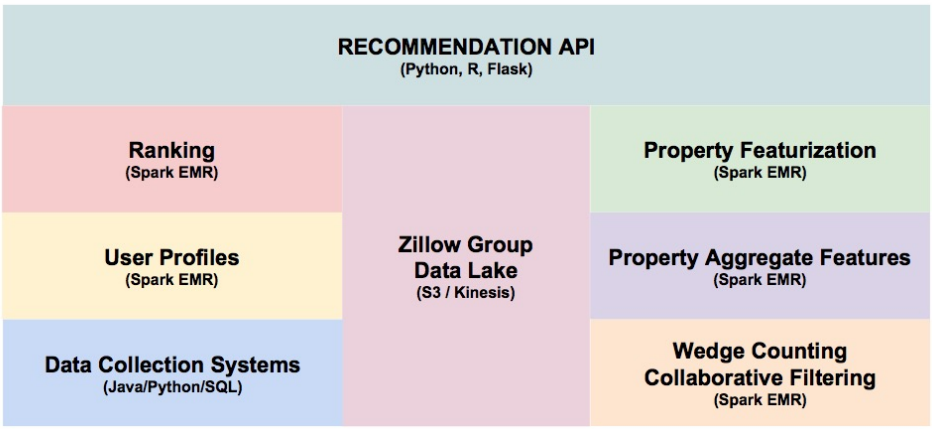


Figure 2: Zillow/Trulia Recommendation Architechture

The recommender system at Zillow and Trulia is based on the same model 'Wedge Counting'. Wedge Counting methodology for recommendation was published by Joseph S. Kong et all []. The process followed for recommendation is : Predict homes (likes/dislikes) based on the behaviour of similar users. Wedge count method is used for all user-property pairs to predict user likes/dislikes. Based on the like/dislike counts, the user-item preference is classified using Gradient Boosting Classifier (sklearn). In the next step, user profiling is carried out based on collaborative-filtering method. The output of these two step is a property matrix which is ranked based on the final scores received.

6.2 Proposed Approach

Based on the problems identified, in this project we aim to try different methods like collaborative filtering , content based filtering and a model adopted by the top real estate website, square counting. We also propose to try a new method which is based Empirical Data Analysis.

6.2.1 Traditional Approaches to Recommender Systems

The traditional recommendation problem can be formulated as follows: given a set of users (or customers) C and a set of items (or services) S with or without associated user/item features and a set of interactions between users and items (users' ratings on items or transactions of user-item pairs), predict the exact or relative utility of individual items for individual users. The prediction can be made using a:

- **Heuristic-based** approach of assuming that the item has similar utility to similar users or
- **Model-based** approach to build predictive models, such as classification models for each item.

Content-based recommendations

Content-based filtering, also referred to as cognitive filtering, recommends items based on a comparison between the content of the items and a user profile. The content of each item is represented as a set of descriptors or terms, typically the words that occur in a document. The user profile is represented with the same terms and built up by analysing the content of items which have been seen by the user. The information source that content-based filtering systems are mostly used with are text documents. The efficiency of a

learning method does play an important role in the decision of which method to choose. The most important aspect of efficiency is the computational complexity of the algorithm, although storage requirements can also become an issue as many user profiles have to be maintained. The learning methods applied to content-based filtering try to find the most relevant documents based on the userâĂŹs behavior in the past. Such approach however restricts the user to documents similar to those already seen. This is known as the over-specialization problem.

Collaborative filtering recommendations

Collaborative filtering, also referred to as social filtering, filters information by using the recommendations of other people. Most collaborative filtering systems apply the so called neighborhood-based technique. In the neighborhood-based approach a number of users is selected based on their similarity to the active user. A prediction for the active user is made by calculating a weighted average of the ratings of the selected users.

References

[1] Felfernig, A., Friedrich, G., Jannach, D., Zanker, M An integrated environment for the development of knowledge-based recommender applications. Int. J. Electron. Commerce 11(2), 11âĂŞ34 (2006a). DOI 10.2753/JEC1086-4415110201

[2] Knijnenburg, B., Reijmer, N., Willemsen, M. Each to his own: How different users call for different interaction methods in recommender systems. In: Proceedings of the ACM Conference on Recommender Systems, pp. 141âĂŞ148 (2011). DOI 10.1145/2043932.2043960

[3] D. Riecken Personalized views of personalization. Communications of the ACM 43(8):26-28, 2000.

[4] Martin P. Robillard, Walid Maalej, Robert J. Walker, Thomas Zimmerman Recommendation Systems in Software Engineering

[5] G. Adomavicius and A. Tuzhilin. G. Adomavicius and A. Tuzhilin. An architecture of e-ButlerâĂŤA consumer-centric online personalization system. International Journal of Computational Intelligence and Applications 2(3):313âĂŞ327, 2002.

[6] G. Adomavicius and A. Tuzhilin. G. Adomavicius and A. Tuzhilin. Using data mining methods to build customer profiles. IEEE Computer 34(2):74âĂŞ82, 2001.

[7] T. Jiang and A. Tuzhilin. T. Jiang and A. Tuzhilin. Dynamic micro targeting: Fitness-based approach to predicting individual preferences. Proceedings of the IEEE ICDM Conference, IEEE Computer Society, Washington, D.C., 173âĂŞ182, 2007.

[8] Plamen Angelov and Xiaowei Gu `EmpiricalFuzzySets`.

[9] Wikipedia contributors. `https://www.marketo.com/definitive-guides/the-definitive-guide-to-web-personalization/`.

[10] Wikipedia contributors. `https://discuss.analyticsvidhya.com/t/what-is-the-difference-between-collaborative-and-content-based-recommendation-system/11113/3`,.

[11] Murray Cox. Inside Airbnb, 1999. Resnick,P.,Varian,H.:RecommenderSystems. CommunicationsoftheACM40(3),56âĂŞ58(1997).

[12] Ingo Feinerer and Kurt Hornik. *tm: Text Mining Package*, 2017. R package version 0.7-1.

[13] Xiaojin Zhu. http://pages.cs.wisc.edu/~jerryzhu/cs769/clustering.pdf.

[14] INFORMS Institute for Operations Research and the Management Sciences

[15] M. Balabanovic and Y. Shoham. Fab Content-based, collaborative recommendation. Com- munications of the ACM 40(3):66-72, 1997.

[16] Francesco Ricci, Lior Rokach,Bracha Shapira,Paul B. Kantor Recommender Systems Handbook, Springer-Verlag New York, Inc., New York, NY, 2010

[17] Arazy, O., Kumar, N., Shapira, B. Improving social recommender

[18] Bailey, R.A. Design of comparative experiments. Cambridge University Press Cambridge (2008)

[19] Golbeck,J Generatingpredictivemovierecommendationsfromtrustinsocialnetworks.In: Trust Management, 4th International Conference, iTrust 2006, Pisa, Italy, May 16-19, 2006, Proceedings, pp. 93âĂŞ104 (2006)

[20] Sinha, R.R., Swearingen, K. Comparing recommendations made by online systems and friends. In: DELOS Workshop: Personalisation and Recommender Systems in Digital Li- braries (2001)