TECHNICAL UNIVERSITY OF CRETE

School of Electrical & Computer Engineering

MSC THESIS

A Novel Hybrid Recommender System for Tourism

Author:
Ioannis Panagiotis Ziogas

Supervisor:
Prof. Georgios Chalkiadakis

A thesis submitted in fulfillment of the requirements for the degree of Master of Science (M.Sc.) in

Electrical and Computer Engineering

June 2023

"Thanks to the people in my life.
A journey that would have never be accomplished without your priceless assistance."

Ziogas Ioannis Panagiotis

TECHNICAL UNIVERSITY OF CRETE

School of Electrical & Computer Engineering

Master of Science (M.Sc.) Thesis

Abstract

A Novel Hybrid Recommender System for Tourism

by Ioannis Panagiotis Ziogas

In this MSc thesis, we put forward several novel recommender algorithms integrated into a hybrid recommender system for the tourism domain.

To this end, we first explore the use of semantic similarity measures for Content-based recommendations to suggest tourist attractions. We study ways of deploying hierarchies of *points of interests (POIs)* and operate upon them with well-known similarity measures originating in the text analysis domain. Then, we progressively build three novel, hierarchy-free, similarity measures and discuss their strengths and weaknesses. We end up with a measure, the *Weighted Extended Jaccard Similarity (WEJS)* that combines information regarding the user interests (in the form of user preference-related weights) and specific item's characteristics (in the form of particular values for the item's features). As such, the use of *WEJS* allows the provision of recommendations that are effectively personalized. Interestingly, though it is a hierarchy-free measure, it is able to recommend items based on others that would naturally appear close in a feature-based POIs hierarchy; while at the same time it is capable of capturing similarities among items that would be distant to each other in any hierarchy build solely based on the POIs' features. Our systematic experimental evaluation using real-world data showcases the benefits and limitations of the various measures and confirms the effectiveness of *WEJS* in offering "rich" and personalized recommendations, so that it can be utilized as important sub-component of a complete Recommender System.

Subsequently, we develop two novel recommender algorithms, a Content-based one and a Hybrid which combines *(a)* a Bayesian component used for eliciting user preferences, and *(b)* the aforementioned Content-based algorithm as recommendations component. The second component can in fact itself be considered a hybrid among two different algorithms exploiting semantic similarity measures: a hierarchy-based and *WEJS*, the non-hierarchy based one. We evaluate our approach via extensive simulations conducted on a real-world dataset constructed for the needs of a real mobile application for short-term visitors of the popular touristic destination of Agios Nikolaos, Crete, Greece. Our experiments verify that our algorithms result in effective personalized recommendations of touristic points of interests; while our final hybrid algorithm outperforms our exclusively Content-based recommender algorithms in terms of recommendations accuracy.

Parts of the research results produced in this thesis appear in three scientific articles, two published after peer-review in a scientific journal and the proceedings of an international conference, and one currently under review.

Πολυτεχνείο Κρήτης
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Διατριβή Μεταπτυχιακού Τίτλου Σπουδών
Ένα Καινοτόμο Υβριδικό Σύστημα Συστάσεων για Τουρισμό

Περίληψη

Στην παρούσα μεταπτυχιακή εργασία, προτείνουμε νέους αλγόριθμους συστάσεων οι οποίοι είναι ενσωματωμένοι σε ένα υβριδικό σύστημα συστάσεων με εφαρμογή στον τομέα του τουρισμού.

Για το σκοπό αυτό, αρχικά διερευνούμε τη χρήση μέτρων σημασιολογικής ομοιότητας για βασισμένες στο περιεχόμενο συστάσεις ώστε να προτείνουμε τουριστικά αξιοθέατα. Μελετάμε τρόπους ανάπτυξης ιεραρχιών σημείων ενδιαφέροντος (POIs) και ενεργούμε σε αυτά με γνωστές μετρικές ομοιότητας που προέρχονται από τον τομέα της ανάλυσης κειμένου. Στη συνέχεια, σταδιακά δημιουργούμε τρεις νέες μετρικές ομοιότητας που δεν αξιοποιούν ιεραρχίες POIs, και συζητάμε τα πλεονεκτήματά τους και τις αδυναμίες τους. Καταλήγουμε σε μία μετρική, την Weighted Extended Jaccard Similarity (WEJS) που συνδυάζει πληροφορίες σχετικά με τα ενδιαφέροντα των χρηστών (με τη μορφή των χρηστών όπου οι προτιμήσεις τους δηλώνονται με βάρη) και τα χαρακτηριστικά των αντικειμένων (με τη μορφή συγκεκριμένων τιμών για τα χαρακτηριστικά του αντικειμένου). Ως εκ τούτου, η χρήση της WEJS επιτρέπει την παροχή συστάσεων που είναι αποτελεσματικά εξατομικευμένες. Είναι ενδιαφέρον ότι, αν και πρόκειται για μία μετρική χωρίς ιεραρχία, είναι σε θέση να συστήνει αντικείμενα που θα ήταν κοντινά εντός μιας ιεραρχίας POI με βάση τα χαρακτηριστικά - ενώ ταυτόχρονα είναι ικανή να καταγράφει ομοιότητες μεταξύ αντικειμένων που θα ήταν απομακρυσμένα μεταξύ τους σε οποιασδήποτε ιεραρχία που δημιουργείται αποκλειστικά με βάση τα χαρακτηριστικά των POIs. Η συστηματική πειραματική μας αξιολόγηση με τη χρήση πραγματικών δεδομένων αναδεικνύει τα οφέλη και τους περιορισμούς των διαφόρων μετρικών ομοιότητας, και επιβεβαιώνει την αποτελεσματικότητα της WEJS στην προσφορά "πλούσιων" και εξατομικευμένων συστάσεων, έτσι ώστε να μπορεί να χρησιμοποιηθεί ως σημαντικό υποστοιχείο ενός πλήρους Συστήματος Συστάσεων.

Στη συνέχεια, αναπτύσσουμε δύο νέους Αλγορίθμους Συστάσεων, έναν Βασισμένο στο περιεχόμενο και έναν Υβριδικό, ο οποίος συνδυάζει (α) ένα Bayesian αλγόριθμο που χρησιμοποιείται για την ανάδειξη των προτιμήσεων των χρηστών και (β) τον προαναφερθέντα αλγόριθμο βασισμένο σε περιεχόμενο ως σύστημα συστάσεων. Το δεύτερο σύστημα μπορεί στην πραγματικότητα να θεωρηθεί το ίδιο ως υβρίδιο μεταξύ δύο διαφορετικών αλγορίθμων που εκμεταλλεύονται μετρικές σημασιολογικής ομοιότητας: έναν Βασισμένο στην Ιεραρχία, και το WEJS – που όπως προαναφέρθηκε δεν βασίζεται σε ιεραρχίες. Αξιολογούμε την προσέγγισή μας μέσω εκτεταμένων προσομοιώσεων που πραγματοποιήθηκαν σε ένα σύνολο πραγματικών δεδομένων που κατασκευάστηκε για τις ανάγκες μιας εφαρμογής κινητών συσκευών για συστάσεις τουριστικών διαδρομών (επικεντρωμένο στις ανάγκες βραχυπρόθεσμων επισκέψεων) του τουριστικού προορισμού του Αγίου Νικολάου στην Κρήτη. Τα πειράματά μας επαληθεύουν ότι οι αλγόριθμοί μας οδηγούν σε αποτελεσματικές εξατομικευμένες συστάσεις τουριστικών αξιοθέατων, ενώ ο τελικός μας Υβριδικός αλγόριθμος υπερτερεί έναντι των αλγορίθμων συστάσεων που βασίζονται αποκλειστικά στο περιεχόμενο όσον αφορά την ακρίβεια των συστάσεων.

Μέρος της παραχθείσας στην παρούσα εργασία έρευνας και των αποτελεσμάτων της εμφανίζεται σε τρία επιστημονικά άρθρα, δυο δημοσιευμένα μετά από κρίση σε επιστημονικό περιοδικό και σε πρακτικά διεθνούς επιστημονικού συνεδρίου, και ένα το οποίο τελεί υπό κρίση κατά την τρέχουσα χρονική περίοδο.

Acknowledgements

I would like to express my deepest gratitude to my professor Georgios Chalkiadakis, who has been a tremendous source of inspiration and guidance throughout my academic journey. His unwavering support and expertise have helped me grow both personally and professionally. I also want to thank my family for their unconditional love and encouragement. Their constant belief in me has motivated me to pursue my dreams and overcome any obstacles. Last but not least, my friends and laboratory colleagues have been my pillars of strength and joy. I am forever grateful to all of you for your presence, but specifically grateful to Lampros, Argyro, Eleytheria and Eva.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Tourism is a widespread activity and a huge industry in our time and era. Almost 900 million visitors traveled abroad in 2022 alone, according to the United Nations' World Tourism Organization. In their travels, tourists have to select among numerous alternatives of landmarks, places and in general *points of interest* (POIs) they can visit. This can be challenging for travelers that have no prior experience of or "inside knowledge" regarding their destination, and even more so for short-term visits. The exhaustive examination of all the possibilities is clearly impossible, thus an information system that can assist them in narrowing their options down to a more tight set would be of much use to tourists. As a result, a booming industry of travel-related *Recommender Systems* (RSs) [72] has been developed, with the purpose of providing recommendations to users that are the most relevant to their preferences. Such systems are embedded in almost every modern mobile tour guide and similar applications.

Based on the methods they adopt, Recommender Systems may be categorized into three main categories, namely the *Content-based* (Cb), *Collaborative filtering* (CF) and *Hybrid* RSs [72, 78, 62]. Nevertheless, they can further be grouped into additional categories, such as *Bayesian Recommenders*, which employ Bayesian updating of user models for efficient personalized recommendations [6, 4, 95]. A recent research work on *travel recommenders* [21] has classified these into mainly three major categories: specifically, hotel, restaurant, and tourism recommenders. The latter can be related to tour planning, group recommendations, touristic attractions-related, or travel packages. Recently, Bayesian recommender algorithms have been combined with social choice-theoretic methods to enable personalized and fair recommendations of touristic POIs, both in single-user and group recommendation environments [91, 92].

Now, although *semantic similarity measures* have been mainly exploited in the domains of text analysis [51] and natural language processing [65], there have also been employed to correlate the preferences of tourists with touristic attractions. All semantic similarity measures describe the relations between different ontologies, and are classified into two categories [29]. The first corresponds to those that generate a hierarchical structure in order to measure similarity and are termed *hierarchy similarity measures*, while the second correlates ontologies without constructing a hierarchy tree and are termed *non-hierarchy similarity measures*.

Against this background, in this work we were eager to investigate the possibility of utilizing semantic similarity measures in our proposed systems. Initially, actual information on user preferences and *points of interests* was gathered by using: *(i)* surveys that visitors completed, *(ii)* internet sources, and *(iii)* local knowledge. In order to employ several well-known hierarchy similarity measures, the items, or *points of interests*, were also organized in a hierarchical structure. Furthermore, we methodically establish three novel similarity measures, that operate in absence of

a hierarchical structure. Among these, we endorse the usage of *Weighted Extended Jaccard Similarity (WEJS)* index, a sophisticated measure that captures the similarity between POIs by importing feature weights and feature values that are influenced by users' interests.

Additionally, we put forward two novel recommendation algorithms: a novel Content-based (Cb) recommender and a *Hybrid* Recommender System that merges the aforementioned algorithm with a Bayesian one. In fact, our Cb recommender algorithm is itself, in some sense, a "hybrid" of two (Cb) approaches that employ two types of semantic similarity measures—a hierarchy and a non-hierarchy one.

The hierarchy similarity measure of choice is the *Extended Wu-Palmer (XWP)* similarity, a variation of the metric proposed in [89], in which POIs lie on the nodes of a hierarchy tree, whereas the other metric is *Weighted Extended Jaccard Similarity (WEJS)* measure (i.e., the novel non-hierarchy similarity measure mentioned above). *WEJS* calculates the correlation between different POIs by considering both the POIs' features values and the user's preferences, thus it produces personalized recommendations. Subsequently, our Cb algorithm combines these two measures in a *hybrid*[1] similarity measures-based method. This Cb algorithm is combined with a Bayesian Inference component that is responsible for building a user model via a lightweight Bayesian elicitation process that asks the user to rate *generic* images (corresponding to generic types of POIs). Then, a set of generic POI items corresponding to the user preferences captured by the Bayesian component, are fed into the Cb algorithm that eventually suggests the actual POIs that are most equivalent to the constructed user model. The high-level structure of our novel *Hybrid RS* (which also includes our novel Cb algorithm) is presented in Figure 1.1.

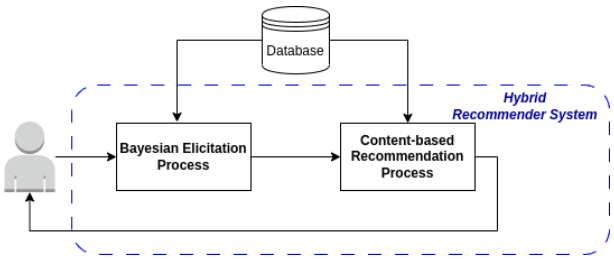


FIGURE 1.1: The high-level structure of our Hybrid Recommender System

As such, the eventual *Hybrid RS* we propose has two desired properties. Specifically, the usage of a probability density-based (usually a multi-dimensional Gaussian distribution)) representation [97] along with *Bayesian Inference* provides us with a formal way to model the high uncertainty that exists in such settings [22], while the Cb technique enables our system to utilize data regarding *(i)* hierarchical structures that contain the available POIs; *(ii)* the "distance" between POIs' characteristics and users' personal interests, via exploiting information contained in user-specific weights assigned to different features.

Now, a common issue arising in recommender systems and especially in Collaborative Filtering techniques is the well-known *cold-start problem*. This refers to the inability of algorithms to provide meaningful recommendations due to the lack of ratings when a new user or item-to-recommend enters the system [27, 100]. This can have a great impact on users' experience when using travel planning applications related especially to short-term travels, since almost all users are new and presumably

[1]The term hybrid here denotes a combination of two similarity measures and it should not be confused with the hybrid recommender algorithm that is an outcome of our work.

interact with the application only a handful of times. Therefore, the (short-term) efficiency of recommender algorithms incorporated in such applications is of utmost importance. In our case, we effectively circumvent and tone down the impact of the cold-start problem, by not employing a CF approach (i.e., by not relying on and comparing with others' ratings) for producing personalized recommendations—but by using a Bayesian updating approach to elicit and maintain user profiles. Specifically, in our work we directly elicit new users' preferences by presenting generic images to them, maintain their profiles as explained in the paper, and make recommendations based on their elicited interests. Avoiding the cold-start problem can be viewed as an advantage of our recommendation approach especially in the context of short-term tourist visits. Of course, quickly arriving at accurate user profiles is a challenge, which can be dealt with effectively via the use of appropriate easy-to-maintain priors [4, 95, 92, 91].

We evaluate our *Hybrid RS* and its components using a real-world dataset from a popular tourist destination, the city of Agios Nikolaos in Crete, Greece. The data related to user preferences was gathered by means of a survey answered by actual tourists; while data on POIs was collected via close collaboration with the local Municipality, other sources of local knowledge, and web sources. Our datasets and algorithms were constructed to be utilized within a real-world mobile app that offers personalized recommendations and tour scheduling services to short-term visitors of Agios Nikolaos and its vicinity. The app (see Figure 1.2), developed in close partnership with the city's Municipal authority, incorporates several recommender algorithms and a deterministic method for the recommendation of personalized itineraries [60], and is currently available at Google Play Store[2].

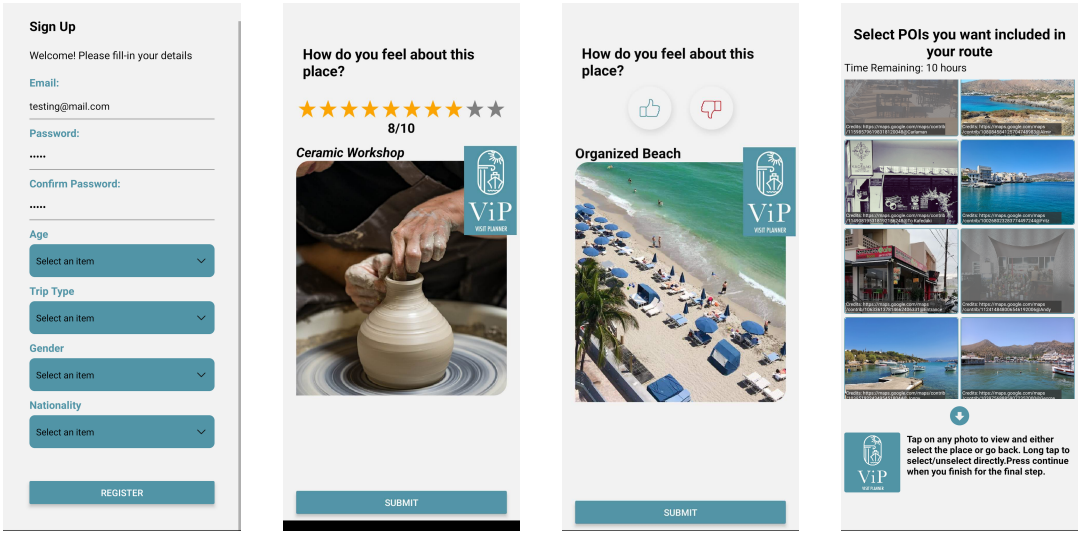


FIGURE 1.2: Our *Visit Planner (ViP)* real-world RS mobile application: Screenshots

The experiments we conducted verify the excellent performance of our RS algorithms, which regularly exhibit a precision of over 83%. In some detail, while our CB recommender components are shown to be quite effective in producing accurate personalized recommendations, our final Hybrid RS demonstrates a superior ability to extract the users' profiles and utilize them in its operation. In particular, its use is shown to result to even more precise personalized recommendations than those generated by the Cb recommenders; and it does so exploiting a *lightweight* Bayesian

[2] https://play.google.com/store/apps/details?id=com.netmechanics.vip&pli=1

elicitation process that requires only a low number of interactions with a user in order to "learn" her profile.

In a nutshell, our work results to several contributions:

1. We construct a dataset and a hierarchical structure of POIs through the collection of real-world data.
2. We develop three novel non-hierarchy similarity measures.
3. We test the similarity measures (i.e., hierarchy-based and non-hierarchy-based) in tourism-related recommendations.
4. We employ *Weighted Extended Jaccard Similarity (WEJS)* for the first time within a recommender algorithm.
5. We combine it with a hierarchy-based similarity measure to give rise to a novel hybrid Cb recommendations algorithm.
6. We combine that Cb algorithm with a Bayesian component to put forward a novel *Hybrid* Recommender System.
7. We evaluate our recommender algorithms via extensive simulations conducted on the real-world dataset, with results testifying to their effectiveness in producing accurate personalized recommendations.
8. Last but not least, our algorithm is already incorporated in a real-world tour planning application used for portable devices for short-term visitors of a popular touristic destination.

It is important to note that the dataset under consideration was constructed through a collaborative effort involving the Technical University of Crete, Hellenic Mediterranean University, and the Municipality of Agios Nikolaos for the specific purpose of facilitating the research project entitled "ViP, Visit Planner: Integrated Information and Tour Planning Service for Cruise Tourism based on Hybrid Recommender System". The dataset comprises various entities, including but not limited to museums, archaeological sites, monuments, restaurants, cafe-bars, islands, and beaches. In addition, the *ViP* research project has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH -CREATE-INNOVATE B cycle (project code: T2EDK-03135). Parts of this master's thesis are reported in three scientific publications, the following:

- Ioannis Panagiotis Ziogas, Errikos Streviniotis, Harris Papadakis, and Georgios Chalkiadakis: Content-Based Recommendations Using Similarity Distance Measures with Application in the Tourism Domain. In Proceedings of the 12th EETN Conference on Artificial Intelligence (SETN 2022), Corfu, Greece, September 2022. [101]
- Georgios Chalkiadakis, Ioannis Ziogas, Michail Koutsmanis, Errikos Streviniotis, Costas Panagiotakis, and Harris Papadakis. "A novel hybrid recommender system for the tourism domain". In: Algorithms 16.4 (2023), p. 215. [20]
- Harris Papadakis, Costas Panagiotakis, Paraskevi Fragopoulou, Georgios Chalkiadakis, Errikos Streviniotis, Ioannis-Panagiotis Ziogas, Michail Koutsmanis, and Panagiotis Bariamis. "Visit Planner: A Personalized Mobile Trip Design Application based on a Hybrid Recommendation Model". In: (2023) (*under review*). [61]

The rest of this thesis is structured as follows. In Chapter 2 we present background and review related work, in Chapter 3 we detail our work on the novel semantic similarity measures we put forward, and present experiments we performed to assess the behaviour of those measures. Chapter 4 is devoted to a detailed description of our Cb recommender system component which combines two different similarity measures for personalized recommendations, while in Chapter 5 we illustrate the *Hybrid* Recommender System which utilizes a Bayesian inference algorithm so as to extract user's preferences before the recommendation process as can be seen in Figure 1.1. Chapters 4 and 5 also include the results of the experiments conducted in real-world data for the proposed recommendation algorithms respectively. Finally, in Chapter 6 we provide a conclusion and future research directions.

Additional information about the research is provided through the appendices. Particularly, the complete hierarchy tree structure that is utilized by the proposed recommenders is depicted in Appendix A, while Appendix B presents the questionnaire that was distributed to tourists in order to obtain their preference-related data. Appendices C and D are devoted to extra experimentation of the proposed algorithm introduced in Chapter 4 and the Bayesian recommender of [91] respectively. Appendix E provides a description about the method we followed for the Twitter-ACM RecSys Challenge 2021.

Chapter 2

Background & Related Work

In this section the theoretical background on Recommender Systems and semantic similarity measures is provided, as well as we briefly review some related work.

2.1 Recommender Systems

Recommender Systems (RSs) form the methods or the software tools creating recommendations for items to be used of a user [72, 17, 52]. Several decision-making processes relate to the recommendations provided by a Recommender System, such as what music to listen, which book to read, which tourist destination to visit, or which items to buy. The term *item* generally denotes what recommendations are being produced to users by the system [72]. Thus, the design, graphical user interface, and core recommendation method of a Recommender System, are all modified in order to generate effective and useful suggestions for a specific type of items (for example travel packages, movies, music tracks and products).

The development of these systems is crucial given the enormous number of the alternative items that a website is able to provide. Primarily, Recommender Systems are destined towards individuals who do not possess the appropriate personal ability and experience to assess all the possible items suggested by a website's catalog. For instance, a music track Recommender System helps individuals to select songs based on their preferences. In the most popular music streaming services provider, Spotify, a Recommender System is employed to create personalized suggestions about music tracks or albums for each subscriber [54]. Given that different individuals or groups of users are provided with various recommendations, the suggestions are mostly personalized. Except from the personalized recommendations, non-personalized recommendations are also available. These are easier to generate with common examples including lists of the top-ten songs, TV-shows, books, etc. Although they could be useful and effective in some circumstances, these types of non-personalized suggestions are not commonly addressed by the Recommender System research [72].

The simplest type of personalized recommendations is the generation of a ranking list of items provided to users [72]. When a ranking list is being created, Recommender Systems seek to predict the most appropriate services (e.g., Spotify's subscribers) or products (e.g., an e-shop's customers) for a user based on her preferences or interests. For the purpose of completing such computational task, the users' preferences are collected, whether they are either inferred by interpreting user actions (i.e., implicit feedback), or are expressed (explicitly) via ratings for items (e.g., services, products). The browsing of a website to a certain service or product, which implies a preference for these displayed items on the page, is a case of implicit feedback. These types of feedback are frequently employed by online retailers (e.g.,

Amazon.com) and collecting this kind of information requires no effort referring to work required by a customer. Utilizing these sources of information to gather user preferences is the basic idea of Recommender Systems. Thus, the recommendation analysis often takes into consideration previous interactions between items and users, as the past preferences often correspond to future choices [2].

The earliest Recommender Systems applied algorithms providing recommendations generated by a group of individuals to provide recommendations to a specific user who was seeking guidance (or suggestions) for a specific catalog of items. The recommended items were were correlated with items that users with similar tastes had liked before. Collaborative filtering is the name given to this approach, and the theory behind it is that if the active user and some other users have previously chosen similar items, the suggestions made by these other users will be relevant to the active user's interests. Many e-commerce websites developed the need to produce suggestions assisting their customers to select wisely from a huge range of alternatives. Users were having a difficult time selecting the best options from the vast diversity of products and services which are available on these kinds of websites.

Recommender systems have recently shown to be an effective technique for addressing the issue of information overload. Eventually, this phenomenon is addressed by recommender algorithms by guiding a user toward new and undiscovered items that may be related to the user's present project. The produced recommendations rely on the usage of a variety of data kinds and information from databases that include knowledge about the accessible items, users, and prior interactions. Therefore, the user has the option to peruse the recommendations, accept them, or reject them and additionally at any stage to provide explicit or implicit feedback. The latest data produced via the user-items interactions and user feedback, are also stored in databases, to be used as knowledge by the recommender techniques in the next user-system interaction.

Recommender Systems arose as a separate study topic in the mid-1990s [3, 31, 55]. Around the world, many institutions of higher education have dedicated to Recommender Systems as courses are now available to undergraduate and graduate students, and many educational materials are now published introducing the different techniques of Recommender Systems, such as the book of Jannach et. al. [41]. The research community appears to be very interested in Recommender System tutorials presented at computer science conferences, such as User Modeling Adaptation and Personalization (UMAP), ACM Special Interest Group on Information Retrieval (SIGIR), and ACM's Special Interest Group on Management of Data (SIGMOD). It is also important to note the annual conference on recommender technology research and applications known as ACM Recommender Systems (RecSys), which was founded in 2007. Furthermore, many highly rated websites such as Youtube, Amazon.com, Netflix, IMDb, Spotify and Tripadvisor have been developing and deploying recommender algorithms in order to present a more user-friendly environment to their customers. For instance, Netflix, the American subscription streaming service for movies and TV shows, offered a million dollar award to the research team that was able to greatly boost the effectiveness of its Recommender System [45].

2.2 Recommender Systems Roles

In this section, we wish to demonstrate the variety of potential roles a Recommender System can play. The authors of Recommender Systems handbook [72] try to distinguish the roles played by a Recommender System on behalf of the service provider

and those of the user of the recommendation algorithm. As for example, a travel Recommender System (i.e., recommendation algorithm for tourism) is usually presented by a organization destination management or travel intermediary to enhance its turnover. While, a user focus on finding suitable hotels and attractions or events based on her interests while touring a tourist destination.

The service providers have several reasons exploiting the recommendation technology explained below [72]:

- **Enhance the user satisfaction:**
  An effective Recommender System can enhance the user's experience. The user will increase system usage and be more likely to accept recommendations if the system is accurate and developed with the right human-computer interaction. Whereas with the term accurate we denote the system to provide interesting and relevant suggestions based on the individual's preferences.

- **Enhance the user fidelity:**
  A website that is frequently visited by a so-called "loyal" user, should recognize that old customer and treat her as an important user. A usual feature of a Recommender System, as many recommender algorithms produce recommendations by enriching data about the user acquired from past user-system interactions, such as the rating of items. As consequence, the user model (i.e., representation of the user preferences by the system) becomes more refined (i.e., the recommender algorithm can effectively customize the recommendation outputs in accordance with the user preferences) based on the duration of the user interaction with the website.

- **Understand more the desires of users:**
  Correctly analyzing user preferences, which might be explicitly gathered or predicted by a Recommender System, is an important characteristic. The service provider may then decide to utilize this type of information once more for a range of goals, such as to improve the management of its product inventory or manufacturing. Based on the information gathered by the Recommender System, destination management companies in the travel industry, for instance, may choose to advertise a certain location to new client groups or a promotional message (i.e., user transactions).

- **Improve the number of sold items:**
  The rise in the quantity of things sold may be the most important aspect in the case of commercial Recommender Systems. A recommender can accomplish this particular purpose since the items they suggest may well satisfy the user's goals and requirements. The objectives are fairly similar in non-commercial recommender apps, when selecting a suggested item is free for the user. For instance, content networks want more people to read news stories on their websites. To put it another way, the main reason for using a Recommender System is to raise the conversion rate, which is calculated by comparing the proportion of users who accept and utilize a recommendation to the total number of visitors who view the content [72]).

- **Sell a variety of items:**
  Another critical function of a Recommender System is to aid users in selecting items that would be difficult to investigate without a specific recommendation. With a movie rental business, for example, a user is interested in finding movies from all potential genres, not just the popular ones. This would be

difficult without a Recommender System since the service provider (e.g., Netflix) cannot afford to take the risk of advertising movies that are unlikely to appeal to a specific individual. As a result, a Recommender System advertises or offers unpopular movies to the appropriate users.

We discussed earlier, why websites which provide services, should employ a Recommender System. However, users may desire a recommender algorithm, if it will successfully assist their goals or work. As a consequence, such system must equalize the needs of these two parties and provide services beneficial for both of them. A classical reference in this field, the paper of Hertloker et. al.[17], denotes eleven significant tasks that can assist a Recommender System in development.

- **Discover good items:**
  Users are given a ranking list of the recommended items along with estimates of how much they will enjoy the recommendations (e.g., rating star scale with values from one to five). A primary task that is addressed by many commercial systems, while in some other cases the predicted rating is not presented.

- **Discover all good items:**
  The recommended items should satisfy some needs of the user. In such circumstances, simply finding good items is unsatisfactory. Particularly when the number of accessible products is limited or the function of the Recommender System is crucial such as in financial and medical applications. In these circumstances, the user may benefit not only from carefully analyzing all of the available options, but also from the rating of the suggested items or additional explanations produced by the Recommender System.

- **Context annotation:**
  Based on a pro-existing context, highlight some items of a list with respect to the user's tastes in long terms. For instance, the worth watching TV shows for a user should be annotated by the employed TV recommender algorithm.

- **Sequence recommendation:**
  Rather than focusing on selecting a single item to recommend, the idea is to recommend a series of items (e.g., the compilation of music tracks [35, 87]).

- **Bundle recommendation:**
  Create a list of items that fit well together. A trip plan, for example, may consist of several attractions, locations, and lodging services available within a specific geographic area. From the user's perspective, these many possibilities may be reviewed and chosen as a specific route [71].

- **Simply browsing:**
  Without any imminent plans to buy anything, the user browses the catalog. The Recommender System's role is to help the user browse through items that are more likely to be relevant to her interests at that specific time.

- **Discover the trustworthy Recommender:**
  Some people experiment with recommenders to see how effective they are at making suggestions since they do not trust recommenders. Therefore, certain systems may offer particular features to enable users to judge its behavior in addition to the activities necessary for getting recommendations.

- **Profile learner:**
  This relates to the user's capacity to enter data about her preferences to the recommender system. A necessary step in the process of providing personalized recommendations. If the system knows nothing about the active user, the recommendations that may provide would be similar with those presented to average users.

- **Expressing opinion:**
  The recommendations might not be of interest to certain users. Their right to participate with their evaluations and express their opinions is what counts to them. User satisfaction for such activity may nevertheless work as a lever to keep the user closely tied to the program, as was mentioned above in examining the service provider's aims.

- **Assist other users:**
  Some people are open to sharing information with the community, such as their opinions on various products. This may serve as a strong incentive to enter data into an infrequently used Recommender System. A user who has previously purchased her new automobile is aware that the rating entered into the system is more likely to be helpful for other users.

- **Affecting other users:**
  There are users in web-based Recommender Systems whose primary purpose is to affect other individuals to purchase specific items. In fact, there are some malevolent people who may manipulate the system to penalize or promote specific items.

2.3 Knowledge & Data Sources

Systems that analyse data and actively gather various sorts of information with the purpose of providing suggestions are known as Recommender Systems. The majority of the information relates to the people who will receive recommendations about items. Recommender Systems may have access to a wide variety of data and knowledge sources, thus whether they can be employed or not ultimately relies on the recommendation approach that is suggested. Typically, knowledge-poor recommendation algorithms employ extremely basic and simple data, such user evaluations (i.e., ratings for things), to make recommendations. Alternative recommendation methods that incorporate ontological descriptions of the items or users, or social connections and user actions, are even more knowledge-intensive. Nevertheless, the information employed by Recommender Systems pertains to three different kinds of objects: items, users, and transactions, that are connections between users and items.

2.3.1 Users

Recommender Systems' users can have a wide range of characteristics or goals. To customize the human-computer interaction and recommendations, Recommender Systems make use of a range of user-related data. The recommendation approach chooses which information to model out of the various ways this information might be arranged. Socio-demographic characteristics like as age, gender, education or occupation are utilized in a demographic Recommender System. The user model is built out of user related data [12, 28]. Subsequently, in the case of Collaborative

filtering, users are portrayed as a straightforward list that includes the ratings that they have produced for certain items.

By capturing the user's interests and requirements, the user model describes her. There have been many different approaches to user modeling, and in some aspects, a Recommender System may be thought of as a tool that generates suggestions by creating and utilizing user models [11]. The user model will always be the most important factor since no personalization is possible without a suitable user model, unless the suggestion is non-personalized, as in the case of the top-10 items. In a Collaborative Filtering approach, for instance, the user is either directly profiled by her evaluations of items or the system generates a vector of factor values from these ratings, with individuals differing in how each component weighs in their models. With a travel recommendation system, users can also be identified based on information about their behavior patterns, such as online browsing habits or trip search habits [52]. User relationships, such as the degree of trust in these relationships, may also be included in user data. A Recommender System may utilize this information to suggest products to consumers that were liked by reliable or similar users.

2.3.2 Items

The objects that are being recommended are referred to as items. Both the value or utility of an item and its complexity may be used to distinguish between them. An item's value might be positive if the user finds it valuable or negative if the consumer chose it mistakenly. Importantly, a user always pays a cost when she buys something, and this cost comprises both the cognitive cost of seeking for the item and the cost that must be paid for it.

For instance, the complexity of a news item which includes important factors, such as the item's representation, time dependent relevant and textual structure, must be taken into account by the creator of the news recommender. Simultaneously, the Recommender System creator must recognize that finding and reading news articles always has a cognitive cost, even if the user is not paying for it. If a chosen item is relevant to the user, the benefit of learning something helpful outweighs the expense; nevertheless, if the item is not relevant, the suggestion of that item to the user has a negative value. Websites, news, books, music tracks, and movies are examples of low complexity and value items. Examples with larger value and complexity are PCs, mobile phones, computers, cameras, etc. Vacation, occupations, financial investments, and insurance plans, are the most complex issues that have been evaluated [59].

According to their technology, Recommender Systems can utilize a variety of item features. Consider a movie recommender, where the genre (e.g., science fiction, comedy, romance, drama, etc.), director, and actors may all be used to classify a movie. This system serves as an example of how the value of an object depends on its features. Items can be expressed using a variety of representational strategies, either as concepts in a domain with an ontological representation or as a single id code in a more basic form.

2.3.3 Transactions

Transactions refer to a recorded interaction between a user and the recommender. They are also recording information that is vital to the system's recommendation algorithm that is produced during human-computer interactions. For instance, a

reference to the user-selected as well as a description of the circumstances surrounding that particular recommendation may be found in a transaction log (e.g., the user query). The transaction may also include explicit input from the user as a rating for the selected item, if it is available.

In reality, ratings are the most typical transaction data type that a Recommender System gathers. These evaluations may be obtained explicitly or implicitly. During the explicit collection of ratings, the user is asked to submit her opinion on an item on a rating scale. Several forms of ratings are presented in the work of [84]:

- **Numerical ratings:**
  An example of numerical ratings is the 1 to 5 stars of Amazon's book recommender.
- **Binary ratings:**
  Binary ratings are selections that require the user to simply choose whether or not a specific item is good.
- **Ordinal ratings:**
  Ordinal ratings (strongly agree, agree, neutral, disagree, strongly disagree), in which the user picks the phrase that best describes her thoughts about a certain item (i.e., via questionnaire).
- **Unary ratings:**
  Unary ratings might imply that a person has viewed, purchased, or reviewed a suggested item favourably. In such cases, the absence of a rating means that there is no proof connecting the user to the item (maybe she purchased it elsewhere).

The system collects implicit ratings in order to derive the user's viewpoint from her behaviors during transactions. For example, a consumer searching for ancient Greece on Amazon.com will be offered with a large range of books. The system "assumes" that the user is enthused about this specific book's type at this point.

Conversational systems have a more detailed transaction model by supporting an interactive process. In these systems, user requests and system actions diverge. In other words, a user may ask for a recommendation, and the system will respond with a list of options. It can, however, request more user options with the purpose of providing better results to the user.

2.3.4 Basic Recommendation Techniques

A Recommender System should be able to predict when a specific item is deserving of recommendation to carry out its primary purpose, which is to identify the helpful (i.e., for the user) items. To do so, the system must be capable of assessing the items' utility destined for suggestion, or at least evaluate certain items by comparing their utility, and then decide which of these are worth suggesting based on that comparison. Even if the prediction step is not clearly specified in the recommendation algorithm, we may use this unifying model to describe the overall operation of a Recommender System.

Consider a simple, non-personalized recommender that merely provides popular music songs to demonstrate the prediction phase of a Recommender System. In the absence of more precise data about the preferences of the user, the justification for using this technique is that a popular music track (i.e., an item that is enjoyed by many users— has high utility) will be liked by an other user rather than some

randomly picked music tracks. As a result, the utility of the popular tracks for this generic user is projected to be rather high.

The Recommender Systems' fundamental models use two types of data:

- attribute information about the users and items (e.g., relevant keywords or textual profiles),
- and user-item interactions (e.g., purchasing behavior or ratings).

Techniques using the latter are known as Collaborative filtering Recommender Systems, whereas techniques using the former are known as Content-based recommenders. It should be noted that while most Collaborative filtering systems employ rating matrices, the model is often centered on the ratings of a single user rather than all users' ratings. Knowledge-based Recommender Systems make suggestions based on explicit user demands. Rather of using historical rating or purchase data, the suggestion is built using external knowledge bases and limits. Some recommender systems blend diverse features to generate hybrids. Hybrid Recommender Systems can utilize the advantages of many types of recommendation algorithms to generate techniques that are more resilient in a variety of settings. These essential models will be addressed briefly in the following sections.

2.4 Content-based Recommender Systems

Finding what a user needs, when she needs it, and in the best way possible has become increasingly challenging due to the volume of material available in digital libraries on the internet and their dynamic and diverse nature. The role of user modeling and individualized information access is therefore becoming more and more crucial. Users today need individualized aid in sorting through the vast volumes of information accessible depending on their preferences and interests. Several information providers utilize a recommender algorithm to customize the information that is shown to consumers [69, 16]. The personalizing procedure for the Recommender System of Amazon.com refers to presenting the proper items for the proper users such as sports equipment for an athlete or programming titles for a programmer [49].

The profile or model containing the user's preferences is constructed by systems that adopt a Content-based recommendation strategy which evaluates descriptions or documents of items rated by that user [58]. As part of the recommendation process, the constructed user profile features are compared to the properties of an item's content. Consequently, a relevance assessment represents the degree of interest that a user shows for a specific item. An information access procedure is significantly more efficient when a profile precisely reflects user preferences. By identifying if a user is interested in a certain website and, if not, preventing it from being displayed, it may, for instance, be used to filter search results.

2.4.1 High Level Architecture

The generation of the user profile (also known as the "user model"), the representation of the items, and methods for comparing the user profile to the item representation are all necessary for Content-based Recommender Systems. Figure 2.1 depicts a Content-based Recommender System's high-level structure. Three phases make up the recommendation process, and each is taken care of by a different component.

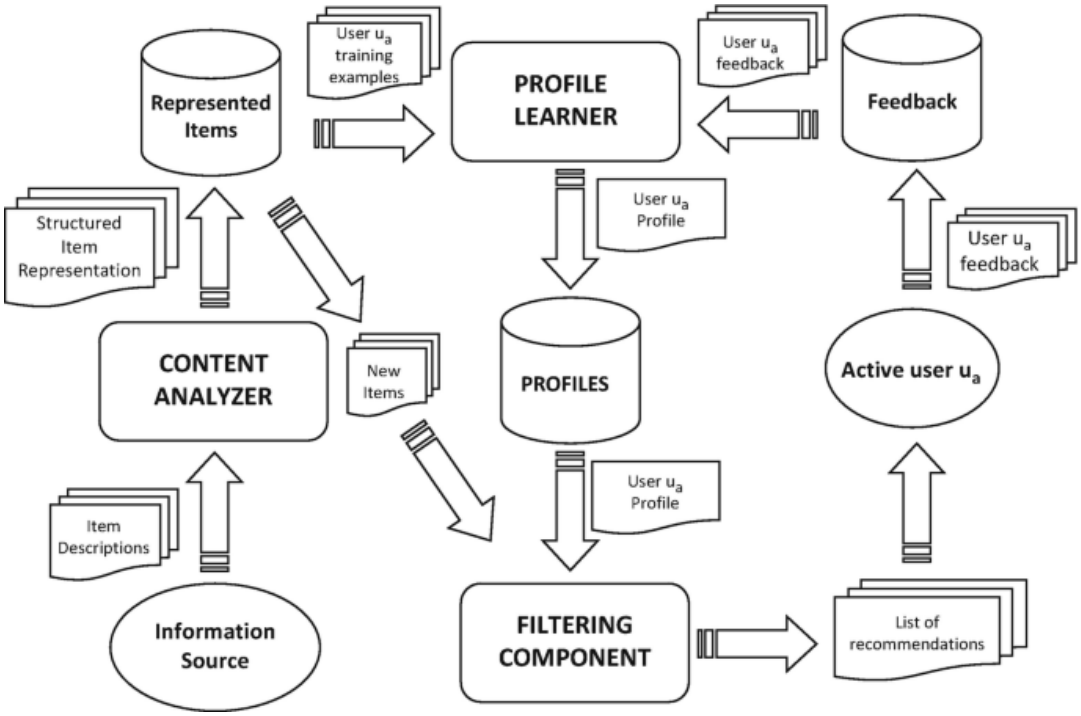


FIGURE 2.1: Content-based recommender's architecture [72]

- **Content Analyzer:**
  Preprocessing is necessary to extract meaningful information that is also structured from material that lacks structure, such as text. The main responsibility of the component is to portray the content of potential items (such as web pages, product descriptions, news, papers, etc.) acquired from information sources in a manner appropriate for further processing procedures. To transform item representation from the source information space to the target one, data items are examined using feature extraction techniques (e.g., keyword vectors representing documents). This representation serves as input for the Filtering Component and Profile Learner.

- **Profile Learner:**
  This module gathers information that is indicative of the user's preferences and makes an effort to generalize it in order to create the user profile. Machine learning techniques are typically used to perform the generalization strategy because they may infer a model of user preferences based on previously liked or disliked items [57]. For instance, a profile learner of a website can utilize a pertinent feedback technique [76] where the learning approach combines vectors of favorable and unfavorable instances into a prototype vector that depicts the user profile. Examples of training include websites that have gotten either favorable or unfavorable comments from people.

- **Filtering Component:**
  The user profile is compared to the representation of the suggested items in this module to propose items that are appropriate for the user. The ultimate result (derived by similarity metrics [36]) is a binary or continuous relevance judgment, where in the second case results in a sorted list of potentially interesting items with respect to the user preferences.

The initial stage of the recommendation process, which frequently uses methods from Information Retrieval systems [5, 82], is carried out by the Content Analyzer. The Content Analyzer builds a structured item representation from unstructured text by extracting features (concepts, keywords, etc.), which is stored in the Represented Items' repository. The responses to things are somehow captured and kept in the Repository Feedback in order to create and maintain the profile of an active user u_a (i.e., person for whom recommendations must be provided). Along with the item descriptions, these annotations/feedback [31] are used to train a model that can forecast the actual relevance of newly presented items. There are typically two categories of relevance feedback:

- negative (or not interesting [37] by the user) data,
- positive (or liked features by the user) data.

There are two possible methods for capturing user input. Explicit feedback occurs when a system requests that the user rate items explicitly. Implicit feedback, on the other hand, occurs when feedback is generated by tracking and analyzing user behaviour without any active user involvement. Explicit assessments highlight a product's relevance to the user [74]. There are three basic methods for obtaining explicit feedback, presented below:

- **Text comments:**
  Similar to [68], feedback from users about a specific item is gathered and provided as a way to build the decision-making process. Purchasers' reviews on Amazon.com, for instance, might help users decide whether a product has earned favorable reviews from other users. Although text comments are helpful, the active user may find them difficult since she must read and comprehend each one to determine whether it is good or negative and to what amount. In-depth approaches from the computer research field are provided in the literature [68] so that Content-based recommenders may perform this kind of analysis automatically.
- **Ratings:**
  Discrete number scales are commonly used to evaluate items, as seen in [88]. The user can assess a website as cold, lukewarm, or hot, or symbolic ratings can be translated to a numeric scale as in [64].
- **Like or dislike:**
  Objects are rated as relevant or not relevant using a simple binary rating system, like in [13].

Despite the potential cognitive load on users and the potential limitations in capturing their item preferences, the use of numeric or symbolic scales may not be a suitable approach for eliciting explicit feedback. In contrast, explicit feedback has the advantage of being a simpler approach. Implicit feedback methods, on the other hand, rely on assigning relevance values to certain user actions on items, such as saving, bookmarking, dismissing, or printing. One key advantage of implicit feedback methods is that they do not require direct user engagement, which can be beneficial in situations where biasing may be possible, such as when phone calls interrupt reading.

Before the active user's u_a profile can be constructed, the training set TR_a must be established. TR_a is a set of pairs $\langle r_k, i_k \rangle$, where r_k refers to the rating provided by

the user u_a for the item representation i_k. The Profile Learner usually utilizes supervised learning techniques to generate a prediction model (i.e., user profile) from a collection of item representations with corresponding ratings. The resulting profile is then stored in a profile repository for later use by the Filtering Component. The Filtering Component determines whether a new item representation is relevant to the active user by comparing the features of the item representation to those of the user preferences representation contained in the user profile. The Filtering Component applies various algorithms to rank potentially relevant items, and the most highly ranked products are presented to the user in a list of recommendations (L_a). In order for the user profile to be updated automatically, up-to-date information must be conveyed to the Profile Learner as user preferences evolve over time. Users may provide feedback on recommendations by indicating their satisfaction or dissatisfaction with products in L_a. After gathering user input, the learning process is repeated using the updated training set, and the resulting profile is customized to reflect the user's current interests. The feedback learning loop is a recurring process that allows the system to account for the changing nature of consumer preferences.

2.4.2 Item Representation

In a Recommender System, a set of properties, features, or attributes are used to represent items that may be recommended to the user. For instance, in a movie Recommender System, the features that are used to characterize a movie include genres, actors, directors, and so on. When an item is described by the same set of properties and the properties have a predetermined set of values, the item is represented by structured data. In such cases, numerous Machine Learning techniques can be applied to learn a user profile [63]. In Content-based filtering systems, item descriptions are textual features derived from sources such as websites, emails, news, and product descriptions. Unlike structured data, textual features do not have predefined properties with set values. The difficulty in learning a user profile with textual features arises from the uncertainty of natural language, which complicates the issue of capturing user interests' semantics. Keyword-based profiles that rely on string matching are unable to capture the semantics of user interests. When a string or morphological variation is detected in both the profile and the document, a match is made, and the document is deemed significant. String matching poses several issues, including polysemy, which refers to the presence of multiple meanings for the same word, and synonymy, which refers to the presence of several words with the same meaning.

In order to overcome the challenges associated with traditional keyword-based methods, semantic analysis and its integration into personalization models have been proposed in the literature. This approach involves the use of knowledge bases such as ontologies or dictionaries to annotate items and describe profiles, allowing for a "semantic" comprehension of the user's information needs. The subsequent section will delve into the basic method for document representation based on keywords.

2.4.2.1 Keyword-based Vector Space Mode

Content-based Recommender Systems typically employ simple retrieval models, such as keyword matching or the Vector Space Model (VSM) with basic TF-IDF weighting. The Vector Space Model is a spatial representation of text documents, where each document is represented as a vector in an n-dimensional space. Each

dimension in the vector denotes a word from the total vocabulary of a document collection, and each document is represented as a vector of term weights, where each weight signifies the strength of the relationship between the word and the document. Here, $D = d_1, d_2, \ldots, d_N$ is a corpus or set of documents, and $T = t_1, t_2, \ldots, t_n$ is a dictionary or a set of words in the corpus. The dictionary is generated using natural language processing techniques, such as stopword removal and tokenization. In an n-dimensional vector space, each document d_j is represented by a vector, with $d_j = w_{1_j}, w_{2_j}, \ldots w_{n_j}$, where w_{k_j} is the weight for term t_k in document d_j. However, the representation of documents in the Vector Space Model poses issues related to term weighting and feature vector similarity measurement. The most commonly used word weighting technique is TF-IDF weighting, which is based on empirical textual findings [81].

- Short documents are preferable over long documents (normalization assumption).
- Multiple uses of a term in a text are just as important as a single use (TF assumption).
- Rare terms are just as important as common terms (IDF assumption).

To state it differently, a term that occurs frequently in a particular document (i.e., high term frequency) but occurs less frequently in the entire corpus (i.e., low inverse document frequency) is more likely to be relevant to the topic of that document. Moreover, to avoid bias towards lengthier documents, normalization of the generated weight vectors is necessary. The TF-IDF (Term Frequency-Inverse Document Frequency) function is an example of the techniques that follow these assumptions:

$$TF - IDF\left(t_k, d_j\right) = TF\left(t_k, d_j\right) \cdot \log \frac{N}{n_k} \tag{2.1}$$

N is the total number of documents in the corpus, and n_k represents the total number of documents in the collection where the term t_k appears at least once.

$$\mathrm{TF}\left(t_k, d_j\right) = \frac{f_{k,j}}{\max_z f_{z,j}} \tag{2.2}$$

The frequency of terms t_k in document d_j is denoted by $f_{z,j}$, and the maximum frequency of any term in d_j is denoted by $max_z(f_{z,j})$. The weight of term t_k in document d_j is then calculated using the TF-IDF function given by Equation 2.1. To standardize these weights, cosine normalization is commonly applied, which scales the weight vectors of each document to have unit length. This ensures that the weights fall in the interval $[0, 1]$ and that all documents are represented by vectors of equal length.

$$w_{k,j} = \frac{TF - IDF\left(t_k, d_j\right)}{\sqrt{\sum_{s=1}^{|T|} \mathrm{TF} - IDF\left(t_s, d_j\right)^2}} \tag{2.3}$$

As mentioned earlier, it is necessary to have a similarity metric to measure the proximity of two documents. One of the widely used similarity measures among the various measures developed is cosine similarity.

$$cos\left(d_i, d_j\right) = \frac{\sum_k w_{ki} \cdot w_{kj}}{\sqrt{\sum_k w_{ki}^2} \cdot \sqrt{\sum_k {w_{kj}}^2}} \tag{2.4}$$

Content-based Recommender Systems that use the Vector Space Model represent both user profiles and items as weighted term vectors. The cosine similarity measure can then be used to predict a user's interest in an item.

2.4.3 Learning User Profiles

Machine learning techniques are commonly employed to generate Content-based profiles, which are well-suited for text categorization tasks [35]. In this approach, a text classifier is automatically created by the machine learning algorithm to categorize text into different categories based on the characteristics learned from a set of training documents labeled with their respective categories. To model user profiles, a binary text classification task is formulated, where each item must be classified as interesting or uninteresting based on the user's preferences. The set of categories $C = c_+, c_-$ consists of two classes: c_+ represents items that the user likes, while c_- represents items that the user dislikes.

2.4.3.1 Naive Bayes & Probabilistic Approaches

Bayesian classifiers, such as Naive Bayes, belong to the class of probabilistic methods for inductive learning. These methods generate a probabilistic model based on historical data.

$$P(c \mid d) = \frac{P(c)P(d \mid c)}{P(d)} \tag{2.5}$$

The Bayesian classifiers, including Naive Bayes, employ a probabilistic method for inductive learning that calculates the a posteriori probability, denoted as $P(d|c)$, of a document d belonging to class c. This probability estimate is derived from the a priori probability, denoted as $P(c)$, which is the probability of a document belonging to class c. Additionally, $P(d|c)$ is the probability of the document d given c, and $P(d)$ is the probability of the instance d. The Bayes theorem is used to determine $P(c|d)$ based on these probabilities. Finally, the class with the highest probability is chosen to classify the document d:

$$c = argmax_{c_j} \frac{P\left(c_j\right) P\left(d \mid c_j\right)}{P(d)} \tag{2.6}$$

Excluding $P(d)$ as it is equal for all c_j, the a posteriori probability of document d belonging to class c, $P(d|c)$, is estimated by observing the training data along with $P(c)$ because its value is unknown. Estimating $P(d|c)$ is challenging because it is highly likely that the same document will be encountered more than once, and the observed data may be inadequate for calculating the appropriate probability. To address this issue, the Naive Bayes classifier simplifies the model by assuming that all words or tokens in the observed document d are conditionally independent of each other, given the class. Instead of estimating the entire document at once, individual word probabilities are calculated individually. The conditional independence requirement is plainly breached in real-world data; yet, empirically, the Naive Bayes classifier performs well in the classification of text documents [12].

Two frequently used models of the Naive Bayes classifier are the multivariate Bernoulli event model and the multinomial event model, as reported in [53]. Despite considering a document as a vector of values over the corpus vocabulary, V, with each element representing whether a word occurred in the document, both models fail to capture information about word order. The multivariate Bernoulli

event model represents each word as a binary property, indicating whether or not the word appeared in the document, while the multinomial event model counts the number of times the word appeared in the document. According to empirical results, the multinomial Naive Bayes formulation performs better than the multivariate Bernoulli model, especially for large documents [53]. The multinomial event model estimates $P(c_j|d_i)$ using its document vector as shown below:

$$P\left(c_j \mid d_i\right) = P\left(c_j\right) \prod_{w \in V_{d_i}} P\left(t_k \mid c_j\right)^{N(d_i,t_k)} \tag{2.7}$$

The number of occurrences of word (or token) t_k in document d_i is denoted by $N(d_i, t_k)$. It is worth noting that only the subset of the vocabulary, V_{d_i}, containing the words that appear in the document d_i, is used instead of the product of all the words in the corpus vocabulary V.

One of the critical steps in implementing Naive Bayes is to estimate the probabilities of words $P\left(t_k \mid c_j\right)$. To make probability estimations more robust with respect to infrequently encountered tokens, a smoothing method is employed. Smoothing is necessary to avoid assigning zero probability values to tokens that are not present in the training data for a particular class. Laplace estimates, which involve adding one to all token counts for a class, is a basic smoothing method.

2.4.4 Advantages & Disadvantages of Content-based Methods

When compared to the Collaborative filtering techniques (i.e., another approach of recommendation algorithms that will further analyzed in next section), Content-based recommenders have various advantages [72],[2].

- **New item:**
  Content-based recommenders are capable of providing recommendations for items that have not yet been reviewed by any user, thereby avoiding the first-rater problem which is often encountered by Collaborative Filtering recommenders that solely rely on user preferences to generate recommendations. In contrast, the Collaborative Filtering systems may not be able to suggest new items until they have been rated by a substantial number of users.

- **User independence:**
  The construction of profiles by Content-based recommenders is solely based on the ratings provided by active users, while Collaborative filtering methods rely on user ratings to determine users who have similar preferences to the active user by identifying nearest neighbors based on their ratings of the same items. The Collaborative filtering techniques subsequently recommend only the most popular items among the active user's neighbors.

- **Transparency:**
  One can provide explanations of the recommended items in Content-based recommender systems by identifying the content features that led to the inclusion of the item in the recommended list. These features serve as indicators for users to assess whether the recommendation is acceptable. In contrast, Collaborative filtering systems are considered black boxes because they provide no explanation beyond the fact that the item was favored by people with similar preferences.

Content-based recommenders also have several disadvantages:

- **New user:**
  To effectively comprehend user preferences and provide accurate recommendations, a Content-based recommender system requires a sufficient number of ratings (i.e., user's feedback). In situations where there are limited feedback available, such as for a new user, the algorithm may not be able to generate dependable recommendations.

- **Limited content analysis:**
  The amount and variety of features that can be automatically or manually correlated with the recommendations made by Content-based approaches has a natural limit. The system must be aware of the actors and directors in order to make movie (i.e., recommendation domain) suggestions, for example. In some cases, domain ontologies are also required. If there is insufficient information in the analyzed content to differentiate between items that the user enjoys and items that they dislike, no Content-based recommender system can provide appropriate recommendations. There are several representations that would impact a user's experience, yet some only capture specific features of the content. To represent a user's interest in drama or comedy, for instance, there is frequently insufficient information in the word frequency, whereas effective computing techniques would be most appropriate.

- **Overspecialization:**
  Content-based recommenders lack an inherent mechanism to provide suggestions for unexpected items. Their recommendations are limited to items similar to those already rated highly by the algorithm. This flaw is known as the serendipity problem, which reflects the tendency of Content-based systems to generate recommendations with limited novelty. For example, when a user has only rated films directed by Quentin Tarantino, only more films of that caliber will be suggested to her. The number of applications for which a very good Content-based technique would be helpful would be constrained because it would rarely discover anything unique.

2.5 Collaborative Filtering Models

Collaborative filtering models utilize the collective power of ratings provided by a large number of users to offer recommendations. However, the sparsity of the underlying ratings matrices poses a fundamental challenge in the development of Collaborative filtering algorithms. For instance, in a movie recommendation system, where users can provide ratings to indicate their preference or disliking of specific films, the majority of users may have only watched a small fraction of the available movies. Consequently, the vast majority of the ratings in the system will be undefined, also known as missing values, while only a small subset of ratings will be defined.

Collaborative filtering approaches rely on the assumption that observed ratings are strongly correlated across different users and items. Therefore, the key idea is to impute the missing ratings in the rating matrix, as a significant portion of the ratings is often undefined. Take two people with pretty similar tastes as an example. The underlying algorithm can determine their similarity if the ratings, which both parties have specified, are quite comparable. In these situations, there is a strong likelihood that the ratings, in which only one of them has assigned a value, will likewise be similar. This resemblance can be utilized to draw conclusions about values that

have not been fully stated. The prediction process in Collaborative filtering typically involves the use of either inner-item correlations or inner-user correlations in most models. Some models use both types of correlations. In addition, certain models utilize optimization techniques to create a training model, similar to how a classifier is trained with labeled data. The missing values in the rating matrix are then filled in using this model, similar to how a classifier predicts missing test labels. Collaborative filtering methods can be categorized into two main types: memory-based and model-based approaches.

- **Model-based methods:**
  In the context of model-based methodologies, machine learning and data mining techniques are employed to build predictive models. The model parameters are learned through an optimization process when the model is parameterized. Examples of model-based techniques include Bayesian methods, latent factor models, decision trees, and rule-based models. Latent factor models, among others, can provide high coverage even for sparse ratings matrices.

- **Memory-based methods:**
  Neighborhood-based Collaborative filtering algorithms are commonly referred to as memory-based algorithms. They estimate user-item ratings based on their neighborhoods and were among the earliest Collaborative filtering algorithms. Neighborhoods can be divided into two categories, namely user-based and item-based.

  1. *Item-based Collaborative filtering:* To generate rating predictions for the target item B by user A in Collaborative filtering, the first step is to identify a set S of items that are most similar to the target item B. User A then provides ratings on the items in set S, which are used to estimate the rating that user A would give to item B. For example, a user's ratings on similar science fiction movies such as Dune and Star Wars can be used to predict their rating on Star Trek. Similarity functions are used to construct relationships between the columns of the ratings matrix, which helps to identify similar items. Memory-based Collaborative filtering algorithms, also known as neighborhood-based algorithms, utilize this approach.
  2. *User-based Collaborative filtering:* In this approach, the ratings provided by users who have similar preferences to a target user A are used to make recommendations for A. The basic idea is to identify users that have comparable preferences to those of the target user A, and then compute weighted averages of their ratings to suggest ratings for the items that A has not rated. This allows us to predict A's ratings on items that have not been rated by A. For instance, if two users have similar preferences for movies, one can use the ratings of the second user on the movie "Dune" to predict the ratings of the first user on the same movie. Typically, the k most similar users to a user are used to estimate that user's rating. To find similar users, similarity functions are computed between the rows of the ratings matrix.

Memory-based approaches are characterized by their ease of implementation and the simplicity of the recommendations generated. However, they have limitations when dealing with sparse ratings matrices, where it may be challenging to identify users who have rated a particular item, such as Dune, and are relatively similar to the target user. In such cases, predicting the target user's rating of Dune is

difficult, resulting in incomplete coverage of rating prediction. Despite this, the lack of coverage is not usually a significant issue when only the top-k items are required.

While memory-based Collaborative filtering algorithms are known for their simplicity, they are heuristic in nature and may not perform well in all scenarios. However, the differentiation between model-based and memory-based techniques is quite arbitrary as memory-based techniques can also be considered similarity-based models.

2.5.1 Rating Types

The design of Recommender Systems is affected by the rating system used to track ratings. The scale used to express ratings may reflect the degree of liking or disliking of the object in question. In some cases, ratings can take continuous values, such as in the recommendation algorithm proposed in [32], where ratings can take any value within the range of $[-10, 10]$. However, such cases are uncommon. Typically, ratings are interval-based and consist of a finite set of ordered integers that define levels of liking or disliking. These ratings are known as interval-based ratings. For example, a 5-point rating scale may be chosen from the set $-2, -1, 0, 1, 2$, where a rating of -2 indicates a strong dislike and a rating of 2 indicates a strong liking for the item. Other systems may use a rating scale of $1, 2, 3, 4, 5$.

The quantity of available ratings may differ based on the particular system. The use of rating scales with 5, 7, or 10 points is particularly common. As an illustration of interval ratings, Figure 2.2 portrays a 5-star rating system. Netflix, for instance, initially utilized a 5-star rating system, with the central point of 3 stars indicating "liked it" and the 4-star point indicating "really liked it." However, since 2017, the user rating has taken on a new form, with "thumbs up" and "thumbs down" as shown in Figure 2.3 [1].

FIGURE 2.2: 5-star rating system example [2]

The use of ordered categorical values, such as "strongly disagree", "disagree", "neutral", "agree", and "strongly agree" (as shown in Figure 2.4), can achieve the same goals. Ordinal ratings are based on the concept of ordinal attributes and are commonly used to express such ratings. In binary ratings, the user can only express a preference or disliking for the item. The ratings could be either 0 or 1, or they might be unspecified. Predictions must be made for undefined values to obtain $0-1$ ratings.

Unary ratings are a type of rating that permits users to indicate their liking for an item but not to express their dislike for it. This type of rating is frequently found in datasets that include implicit feedback, as reported in [39, 38]. In such situations, customer preferences are inferred from their actions rather than from explicitly provided ratings. For example, a customer's purchasing behavior can be transformed

[1] The figure was provided by: https://shorturl.at/cpE29.

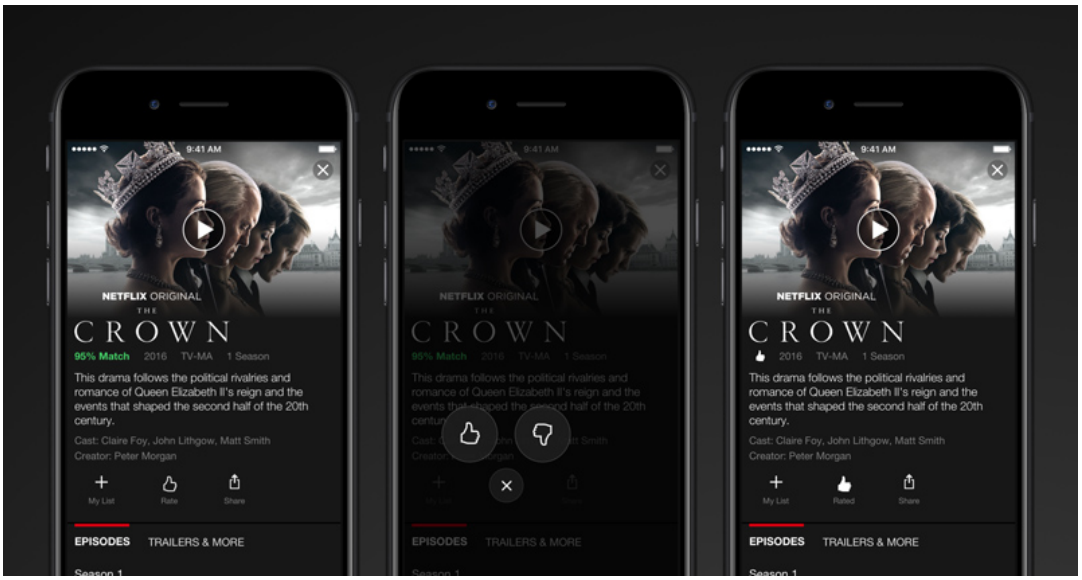


FIGURE 2.3: "thumbs up" and "thumbs down" rating system of Netflix

FIGURE 2.4: Stanford University ordinal rating system for course evaluation [2]

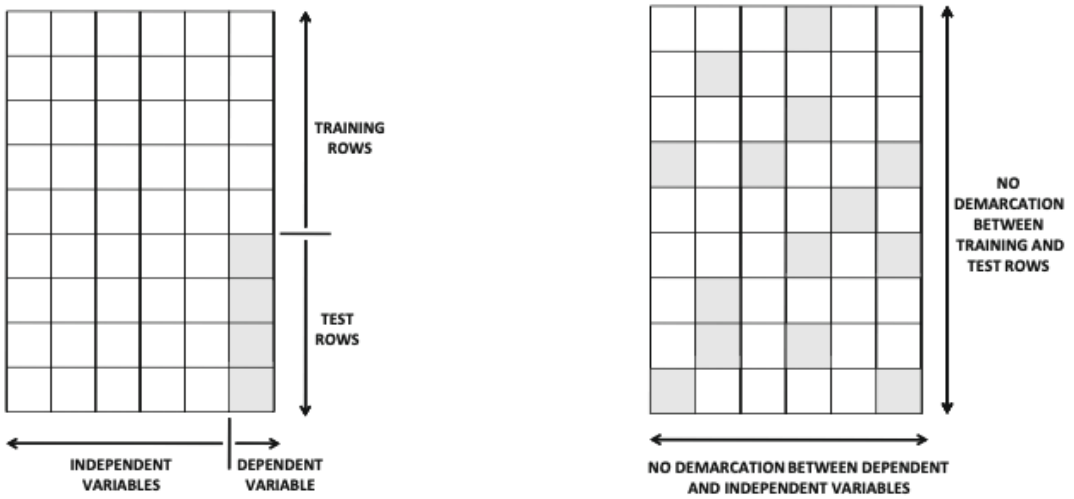
into unary ratings. When a buyer purchases an item, this can be considered a preference for the item. Consequently, the absence of a purchase from a vast array of options does not always indicate dissatisfaction. Similarly, many social networks, such as Facebook, provide "like" buttons that allow users to indicate their preference for an item, but there is no way to express dislike for it.

2.5.2 Model-based Approaches

In model-based methods, similar to supervised or unsupervised machine learning approaches, a condensed representation of the data is constructed initially. Because of this, the phases of training (or developing models) and prediction are clearly separated. Traditional machine learning approaches that use Bayesian classifiers, latent factor models, decision trees, regression models, rule-based methods, and neural networks are examples of such methods [1]. Surprisingly, practically all of these models are applicable to the Collaborative filtering scenario. Given that traditional regression and classification tasks can be considered as special cases of matrix completion problem or Collaborative filtering problem, they are inherently connected.

In the data classification problem, we consider a matrix of size $m \times n$. The first $n-1$ columns represent the feature variables (or independent variables) and the last column represents the class variable (or dependent variable). Only a subset of the entries in the last column are given, while all the entries in the first $n-1$ columns are fully specified. The subset of matrix rows that are fully specified is considered as the training data, while the remaining rows constitute the test data. For the test data, the missing entries need to be inferred. The scenario is illustrated in Figure 2.5a, where

the shaded entries represent the missing values in the matrix. The problem is related to matrix completion and Collaborative filtering, as the task involves predicting the missing entries in a partially observed matrix.



(A) Classification (B) Collaborative filtering

FIGURE 2.5: The classification problem is compared to the Collaborative filtering problem, where in both cases the shaded elements must be predicted [2].

In contrast to the data classification problem, the matrix completion problem may contain missing entries in any location of the matrix, as illustrated by the shaded entries in Figure 2.5b. As a result, the matrix completion problem can be considered as a generalization of both regression modeling and the classification problem. The following distinctions between these two scenarios can be outlined:

- In the data classification problem, a clear differentiation is made between the training and test data sets. However, in the Collaborative filtering scenario or matrix completion problem, the separation between matrix rows, which are used to construct training and test data sets in classification problems, is not present. The specified entries in the matrix are considered as the training data, while the missing entries, which are undefined or unspecified, can be viewed as the test data.

- The data classification problem involves a clear distinction between independent feature variables and dependent class variables. However, in the Collaborative filtering scenario, this division is not apparent. Depending on the predictive modeling's specific entries, each column can serve as both a dependent and independent variable.

- In data classification, columns signify variables and rows denote data instances. However, the Collaborative filtering problem has a unique missing entry distribution, which enables similar treatment of both the original ratings matrix and its transpose. For instance, user-based neighborhood models can be seen as straightforward extensions of nearest neighbor classifiers. When implemented on the transpose of the ratings matrix, they are referred to as item-based neighborhood models. Typically, different classes of Collaborative filtering algorithms have both user-based and item-based variations.

Figure 2.5 depicts the differences between Collaborative filtering and data classification. Compared to data classification, the Collaborative filtering problem's higher generality results in a larger number of algorithmic solutions.

In the design of learning algorithms for Collaborative filtering, it is important to recognize the similarities with the data classification scenario. The well-established nature of data classification and the numerous available classification solutions provide valuable guidance for the development of Collaborative filtering systems. Indeed, most machine learning and classification techniques have direct counterparts in the Collaborative filtering literature. The utilization of various meta-algorithms from the classification literature permits recommender systems to be viewed similarly to classification models [2]. Ensemble-based approaches, such as boosting, bagging, and model combination, can be extended to Collaborative filtering, similar to the classification literature. Notably, much of the theory for ensemble methods in classification applies to recommender systems as well, and these methods were among the top performers in the Netflix challenge [44].

The straightforward generalization of data classification models to the Collaborative filtering problem is often challenging, particularly when a large proportion of the entries are missing. Moreover, the effectiveness of different models varies depending on the situation. For instance, modern Collaborative filtering models, such as latent factor models, are highly suitable for Collaborative filtering. In contrast, such models are not considered competitive in the context of data classification.

A multitude of ways demonstrate that model-based recommender systems often exceed the performance of neighborhood-based methods [2],[72].

- **Training and prediction speed:**
  The quadratic pre-processing stage of neighborhood-based approaches is a disadvantage that scales with either the number of users or the number of objects. In contrast, model-based systems typically have faster preprocessing phases when building the trained model. The summarized model produced during this phase is often sufficient for generating accurate predictions.

- **Avoid the overfitting:**
  The phenomenon of overfitting, where the prediction is excessively influenced by random artifacts in the data, is a common problem in various machine learning algorithms, including classification and regression models. Model-based techniques employ summarization to prevent overfitting. Moreover, these models can be reinforced with regularization approaches.

- **Efficiency of space:**
  The size of the learned model is typically significantly smaller than that of the initial ratings matrix, resulting in relatively low space requirements. However, a user-based neighborhood method, where m is the number of users, may have a space complexity of $O(m^2)$. On the other hand, an item-based method has a space complexity of $O(n^2)$.

Neighborhood-based approaches were some of the first collaborative filtering techniques, and because of their ease of use, they were also among the most popular. Notwithstanding, model-based techniques are not necessarily the most precise models at present. In fact, some of the most precise techniques rely on model-based approaches in general and, specifically, on latent factor models.

2.5.2.1 Latent Factor Models

Latent factor models are a class of models that include neural networks [80], Latent Dirichlet Allocation [14], and models obtained through factorization of the user-item ratings matrix, such as SVD-based models. The popularity of matrix factorization models has recently increased due to their accuracy and scalability. SVD has been widely used in information retrieval to identify latent semantic factors [25]. However, applying SVD to explicit ratings in Collaborative filtering is challenging due to the significant amount of missing values in the matrix. Traditional SVD is undefined when the matrix is incomplete, and overfitting is highly probable when addressing only the known entries carelessly. Imputation has been used to fill in the missing ratings and increase the density of the rating matrix, but it can be computationally expensive and can significantly impact the data due to improper imputation. Some works [43, 18, 7] have proposed modeling using only the observed ratings directly while avoiding overfitting with a suitable regularized model.

2.5.3 Neighborhood-based Approaches

One of the first algorithms created for Collaborative filtering was a group of algorithms known as memory-based algorithms or neighborhood-based Collaborative filtering algorithms. These algorithms are based on the notion that users with similar characteristics have comparable rating patterns, and items with similar attributes receive similar ratings. Neighborhood-based algorithms are classified into two types:

- **Item-based Collaborative filtering:**
  To generate recommendations for a target item B, the initial step is to select a set S of items that are most similar to item B. The ratings for the items in set S, provided by user A, are utilized to predict the rating of any user A for item B. The anticipated rating of user A for item B is then determined by computing the weighted average of these ratings.

- **User-based Collaborative filtering:**
  The recommendations for user A are generated by leveraging the ratings given by other users who exhibit similar rating behaviors to A. Specifically, the ratings given by the group of similar users to user A are aggregated by computing the weighted average of the ratings for each item, which are then used to predict the ratings that user A would assign to those items.

The user-based Collaborative filtering and item-based Collaborative filtering algorithms exhibit marked differences. In the former approach, ratings are projected by the ratings of similar users, while in the latter approach, ratings are projected using the user's own ratings on closely related items. To elaborate, neighborhoods in the user-based approach are determined by the similarity among users (i.e., rows of a ratings matrix), while neighborhoods in the item-based approach are determined by the similarity among items (i.e., columns of a ratings matrix) [2]. These approaches have a complementary relationship; however, the recommendations obtained through these methodologies differ significantly.

Assuming an incomplete m x n user-item ratings matrix $R = [r_{ui}]$, where m represents the number of users and n represents the number of items, it can be inferred that only a portion of the ratings matrix is either observed or defined. To design Neighborhood-based Collaborative filtering algorithms, there exist two distinct methods:

- **Finding the top$-k$ items or top$-k$ users:**
  Typically, the merchant is not concerned with specific rating values for item-user combinations. Instead, they are interested in identifying the top$-k$ most similar items for a given user or the top$-k$ most similar users for a particular item. It is worth noting that finding the top$-k$ products is more common than finding the top$-k$ users since the former scenario is frequently employed in web-centric domains to provide recommended item lists to users. Traditional Recommender Systems primarily aim to discover the top$-k$ items, whereas the latter formulation is valuable to the merchant because it can help identify the most suitable individuals to target with marketing efforts.

- **Prediction of the ratings of user-item combinations:**
  This is the most basic and simplest type of a Recommender System. The prediction of an undefined rating (r_{ui}) of an item i by a user u is being made.

2.5.3.1 Main Advantages of Neighborhood-based Methods

The main advantages of neighborhood-based Collaborative filtering algorithms are [72]:

- **Efficiency:**
  One of the primary advantages of neighborhood-based Collaborative filtering systems is their effectiveness, which does not require expensive training phases, unlike model-based systems. Even though the recommendation phase is often more expensive than model-based techniques, the nearest neighbors can be computed offline, allowing for immediate recommendations. Additionally, these nearest neighbors occupy very little memory, making such systems scalable for applications with millions of users and items.

- **Justifi-ability:**
  These techniques offer an intuitive justification for the computed predictions. For example, in item-based recommendation, the list of neighbor items and their respective ratings assigned by the user can be presented to the user as a rationale for the recommendation. This can aid the user in better understanding the recommendation and its significance and could form the basis for an interactive system in which users can select which neighbors should be given priority in the recommendation [8].

- **Simplicity:**
  Neighborhood-based approaches are generally simple to implement. In their most fundamental form, only one parameter (i.e., the number of neighbors utilized in the prediction) needs to be adjusted.

- **Stability:**
  Another favorable characteristic of these systems is their immunity to the constant influx of ratings, items, and users, which is a common occurrence in large commercial applications. Additionally, once a few ratings for a new item have been provided, only the similarities between this item and those currently in the system need to be calculated.

2.6 Recommender Systems Evaluation

There are several similarities between Collaborative filtering evaluation and classification evaluation, stemming from the fact that Collaborative filtering is a generalization of regression and classification modeling problems. However, numerous components of the evaluation process are specific to Collaborative filtering systems. Due to the fact that Content-based approaches often use text classification methods beneath the surface, their evaluation is even more analogous to that of classification and regression modeling. Properly designing the evaluation system is critical to comprehending the effectiveness of various recommendation techniques. A faulty experimental evaluation design can result in a significant underestimation or overestimation of the actual accuracy of a given recommendation model.

The evaluation of Recommender Systems can be performed using either online or offline methods. In online evaluation, the responses of users to the recommendations are analyzed. Therefore, user engagement and interaction are crucial in online evaluations. For instance, in an online evaluation of a news recommender system, the click-through rate of users on the recommended articles can be used as a measure. A/B testing is a specific type of online evaluation that directly measures the impact of the Recommender System on the users. Offline evaluation, on the other hand, does not involve user interaction and is conducted on a pre-collected dataset. The accuracy of the system is measured based on how well it predicts the actual ratings of users for items that they have already rated. The offline evaluation is commonly used to compare different algorithms and to tune their hyperparameters. A proper selection of evaluation metrics is essential for both online and offline evaluations in order to obtain a reliable measure of the system's performance. A flawed evaluation design can lead to incorrect conclusions and hinder the development of effective Recommender Systems.

The primary goal of a recommender system is to increase the conversion rate of profitable goods, which is an important measure of its effectiveness. However, evaluating a recommender system using online methods, which depend on user interactions, may not be suitable for benchmarking and research purposes. It is often challenging to obtain user conversion data from large-scale systems that interact with many users. Additionally, if access is granted, it is usually limited to a single system.

To achieve a stronger generalization capacity, it is common to desire to use datasets from different domains and of various types. Utilizing numerous datasets is critical to testing the Recommender System's performance in diverse contexts. In such cases, offline evaluations are commonly used. For evaluating Recommender Systems, offline methods are the most popular in both research and practice.

The evaluation of Recommender Systems through offline techniques is not fully represented by accuracy metrics alone, and therefore other measures are also significant. In order to reflect the performance of the Recommender System from the user's perspective, it is essential to design the evaluation system with care, taking into account additional metrics. The following challenges are particularly significant when building Recommender System evaluation methods:

- **Accuracy metrics:**
  The evaluation process of recommender systems places significant emphasis on accuracy metrics. These metrics can be used to evaluate the accuracy of the predicted ratings or the accuracy of the recommended item rankings. The mean error and mean squared absolute error are common measures used

for rating prediction accuracy evaluation. Meanwhile, rank-correlation coefficients, utility computations, and receiver operating characteristic curve are among the different approaches for ranking evaluation.

- **Goals of evaluation:**
  Although accuracy metrics are commonly used to evaluate recommenders, they may not always provide a comprehensive view of the user experience. In addition to accuracy, other factors such as trust, coverage, novelty, and serendipity also contribute to the user experience and can impact short and long term conversion rates. However, measuring these characteristics can be subjective and there may not be standardized metrics available to quantify them numerically. Nevertheless, it is important to consider these factors in the design of evaluation systems to provide a more comprehensive understanding of the performance of Recommender Systems.

- **Experimental issues:**
  The proper design of experiments is crucial to ensure that accuracy metrics are not underestimated or overestimated. If the same ratings are used to construct the model and evaluate its accuracy, the accuracy will be overestimated. Therefore, it is essential to employ a careful experimental design to avoid such an outcome.

2.7 Recommender Systems Applications

This section provides a brief overview of the various applications of Recommender Systems and their specific goals. Table 2.1 summarizes the recommended items and the objectives achieved by different implementations of Recommender Systems. While most of these systems are primarily utilized for product recommendations in e-commerce, Recommender Systems have also expanded into other domains. It is important to note that some of the systems listed in Table 2.1 may not recommend specific products, such as the Google search application that includes product advertisements in its search results. This application belongs to the field of computational advertising, which is closely related to Recommender Systems. Additionally, Facebook suggests friends, while online recruitment platforms recommend employers and job seekers, both of which are examples of reciprocal recommenders. Some of these recommendation algorithms employ models that are different from standard Recommender Systems.

System	Product Goal
Facebook	Friends, Advertisements
YouTube	Videos
Tripadvisor	Travel Packages/Products
Netflix	Streaming Video
Spotify	Music
Google News	News
IMDb	TV Shows/Movies

TABLE 2.1: Products for recommendation by real-world recommenders

2.7.1 Netflix Recommender System

Netflix[2] initially started as a DVD rental service for movies and TV shows through mail-order, later expanding to streaming delivery. Its main business model currently involves subscription-based streaming of movies and TV shows. Users can rate movies and TV shows on a 5-point scale, and Netflix stores their viewing activity as well. Based on these ratings and behaviors, Netflix provides suggestions for users. Additionally, it provides a clear explanation of the recommended items, presenting examples of recommendations based on specific items that the user has viewed.

The user is presented with meaningful explanations about why they might find a particular video interesting, allowing them to make informed decisions about whether or not to view it. This approach enhances the user experience and increases the likelihood of the user acting on the recommendation, which, in turn, can increase customer loyalty and retention. Netflix's participation in the Netflix Prize competition [9], which was designed for participants to compete with their collaborative filtering algorithms, made a significant contribution to the research community. The competition required users to predict ratings of specific user-item combinations using a data set of Netflix movie ratings, with the training set containing 100,480,507 ratings from 480,189 users and 17,770 movies, and a smaller probe set of 1,408,395 ratings that were more recent and statistically matched the hidden ratings in the data set. The qualifying data set contained over 2,817,131 triplets, and users had to predict the ratings in the qualifying data set using training data models, with only half of the qualifying data set being used for this prediction. The prediction was scored by judges, and the results were displayed on the leader board, with the second half of the qualifying data set being used as the quiz set and the remaining half used to compute the final score and determine the prize winners. The test set's unusual configuration was designed to prevent users from overfitting their algorithms to the data by using the leader board's rankings as a justification.

The probe set, quiz set, and test set were intentionally designed to have similar statistical characteristics. Rewards were granted to participants who enhanced Netflix's existing recommendation algorithm, Cinematch, or achieved a score surpassing a specified threshold over the previous best score. The Netflix Prize competition served as a platform for many well-known recommendation algorithms, such as latent factor models, and had a significant impact on recommendation research, as evidenced by several contributions [50].

2.7.2 Twitter-ACM RecSys Challenge 2021

Twitter comprises a vast amount of information on worldwide discussions and events. The curation and dissemination of appropriate information to users is a critical process. Twitter users are presented with two types of tweets in their home timelines, those from accounts they follow and those deemed relevant to them. Users may also determine the order of tweet appearance in their timeline, or the order may be determined algorithmically. In the latter, a predictive model scores each tweet based on its perceived interest or engagement, and the order is established accordingly. The aforementioned ranking model plays a crucial role in user experience, as it has the potential to impact the home timeline and tweet sequence of each user.

[2] https://www.netflix.com

The objective of the ACM RecSys Challenge 2021[3], as organized by Twitter[4], is the prediction of user engagement. Tweet engagements, which include liking, replying, retweeting, and retweeting with a comment (quoting), are the focus of this challenge. Participants are expected to conduct investigations into the various features of the data and use techniques such as feature engineering and classification to train models for each type of engagement. The task is framed as a binary classification problem utilizing gradient boosting trees. Finally, the efficacy of models is evaluated using two metrics: relative cross entropy (RCE) and average precision (AP) for each type of engagement.

Traditionally, machine learning challenges have prioritized accuracy, ranking results based on this metric. However, a more comprehensive system would also account for other factors, such as fairness. The objective of Twitter's challenge of 2021 was two-fold: first, to predict the likelihood of various types of engagement by a target user for a given set of tweets using input data; and second, to provide recommendations that are equitable. Fairness is a social concept that cannot be fully addressed through optimization techniques alone. Specifically, recommendations should not be influenced by the popularity of Twitter users, and users with lower popularity should not be penalized as a result.

Twitter has released a publicly available dataset consisting of nearly one billion data points, which includes 710.3 million engagements over a 28-day period. The first three weeks of this dataset were designated for training purposes, while the last week was reserved for evaluation and testing. The evaluation test was provided to researchers to evaluate the efficacy of their predictive models, while the test set was exclusively available on a test server for use in the competition. Subsequently, following the conclusion of the RecSys Challenge 2021, the test set was made publicly available for the unconstrained leaderboard rankings.

Participants in the challenge were permitted to construct user engagement prediction models using any of four different types of features:

- Tweet features: These features pertain to the content of the tweet.
- Engagee features: These features describe the user who engaged with the tweet.
- Engager features: These features describe the author of the tweet.
- Engagement features: These features relate to the possible engagements, including the timestamp for each engagement, if applicable.

2.8 Recommender Systems for The Tourism Domain

It is common for individuals to relocate to different geographic locations during their lifetime. This type of relocation, also known as a journey or trip, may involve frequent movement within local areas or more extended international travel, and may involve temporary lodging accommodations such as hotels and restaurants. Traveling may occur for various reasons, such as business engagements, adventure, events, leisure, migration, research, or simply to satisfy a desire to explore new destinations. Such travelers contribute to the economic growth of the host communities as well as to local and global businesses. Goeldner et al. [30] have defined tourism as "the science, art, and business of attracting visitors, transporting them, accommodating them, and graciously catering to their needs and wants." In recent years, the

[3] https://recsys.acm.org/recsys21/challenge/

[4] https://twitter.com/ACMRecSys

tourism industry has expanded significantly, and numerous travel services are now available both physically and online.

Gunn et al.[34] proposed five functional groups for tourism development, namely, attractions, services and facilities, transportation, information and direction, and tourists. The tourism industry involves many service providers that have studied these aspects. However, the increasing number of service providers makes it challenging to select the most appropriate travel plan. Recommender Systems have been created to assist individuals in identifying the best travel plan with minimal effort. Recommender Systems belong to a subclass of information filtering systems that can anticipate a person's preferences and recommend suitable options[73]. These systems can present recommendations that may be of interest to a particular user [69].

In the past, conventional travel plans were usually selected based on suggestions from acquaintances or travel agents. Nowadays, however, people have a plethora of digital resources at their disposal. By utilizing others' reviews, photos, ratings, blogs, and other publicly available information, people can assess potential destinations in terms of the journey, accommodations, cuisine, or events. However, as the number of digital sources has increased, manually analyzing available options has become more challenging and time-consuming. This has elevated the expectations for personalized recommendations, in which a system is anticipated to learn about the user's preferences by analyzing various aspects based on implicit and explicit feedback, and offer helpful suggestions. These recommendations can pertain to various travel-related services, such as lodging, dining, or destinations to visit, as well as activities to partake in, foods, drinks, and other amenities [21]. For instance, a user who has visited several heritage sites in the past may be suggested to visit a museum as an example of personalized recommendation.

The domain of electronic tourism (e-tourism) has emerged as a new platform for travel-related advice. E-tourism enables interested users to ask queries about tourist destinations, travel packages, expenditure details, or time constraints, and recommend a list of points of interest (POIs) that match their preferences, thereby emphasizing the importance of personalizing tourism. To address the diverse aspects of travel-related recommendations, numerous researchers have investigated various aspects of the field. For example, Chaudhari et al. [21] analyzed different categories of Recommender Systems used in the tourism domain. They first surveyed the recommenders responsible for restaurant and hotel recommendations, then examined the various features responsible for the selection of respective modules such as hotels and restaurants, and the significance of such features. They also provided details on tourism packages, planning, and group recommendations, along with associated features, and discussed POIs corresponding to tourist attractions, museums, and other events. Finally, they analyzed the widespread considerations involved in a tour, such as photography, outfits, food, climate, safety aspects, and other itinerary recommendations.

2.9 Semantic Similarity Measures

The semantic similarity, or semantic distance, between two concepts, given an ontology, measures the similarity of the concepts based on their common characteristics [90]. As such, it is mainly used in order to present the ontological relationships between words in cases of similarity identification between documents. The structure definition of the compared concepts, which are also related, is formally called IS-A hierarchy. In general, the most developed semantic similarity measures are

those applied to the taxonomic or ontological representation of the context [33]. Various types of hierarchical data are represented by tree structures such as documents, genealogies, catalogs, language corpus etc. Based on this idea, in the following chapters we work with such measures as well as we develop some novel semantic metrics with the purpose of employing them in the recommendation domain, as the best of our knowledge such approaches are no-existent in Recommender Systems.

In more detail, given an ontology, similarity between objects can be established either as the distance between individual concepts or as the distance between sets of concepts. The semantic similarity between two (individual) concepts can be determined by: *(i)* the edge-based approach and *(ii)* the node-based approach. Specifically, edge-based approach relies on applying graph distance measures to a hierarchy tree considered as a directed acyclic graph (DAG); while node-based approach relies on comparing properties of the concepts involved, such as the concepts themselves, their descendants or their ancestors. For instance, [15] introduced an algorithm that calculates the distance between two concepts in a hierarchy by the hierarchy level of their nearest common parent node. Information Content (IC) is used in node-based approaches and provides how informative a node is. An information content-based approach was introduced in [70] by Resnik and computes the relatedness between the concepts as a function of their information content, given by their probability of occurrence in a corpus [93]. On the other hand, the semantic similarity between two sets of concepts is mainly used in computing similarity for information retrieval purposes. Jaccard Similarity [40] is an example of a similarity metric when the sets are represented as vectors in a hierarchical structure. Now we present some well-known semantic similarity measures in the literature.

Definition 2.9.1. *Path length similarity (PLS)* is defined as the length between two concepts which connects them through their *Least Common Ancestor (or Subsumer) (LCA)* [66]. We used the term path loosely because of the hierarchy tree being in fact a DAG. Specifically:

$$PLS(X, Y) = N_1 + N_2 \quad \text{with} \quad X \neq Y \tag{2.8}$$

where N_1 is the number of arcs between concept X and the LCA with concept Y. N_2 is the number of arcs between concept Y and the LCA respectively. Note that the path length similarity between identical concepts will be equal to zero.

Definition 2.9.2. Z. Wu and M. Palmer [98] proposed a similarity metric, known as *Wu-Palmer similarity (WP)*, which describes the semantic similarity among concepts X and Y ($X \neq Y$) as:

$$WP(X, Y) = \frac{2N}{R_1 + R_2} \tag{2.9}$$

where R_1, R_2 is the number of arcs between concepts X, Y and the root node respectively. N is the distance of the LCA for the concepts X, Y from the root node. A similarity distance for two compared objects or concepts of an ontology with a value which is close to 1, indicates a high similarity of these objects. On the contrary, not similar objects (i.e., distant in the hierarchy) have Wu-Palmer similarity near to zero.

Definition 2.9.3. Shenoy et al. [89] developed a new similarity measure for concepts in the same hierarchy by taking into consideration the WP measure. Their inspiration is based on the simplicity and good performance WP similarity can offer. Furthermore, they combined the depth of the whole taxonomy and the shortest path

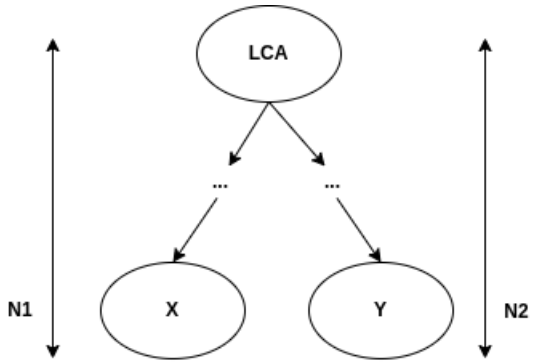


FIGURE 2.6: Example of a concept hierarchy for *PLS*

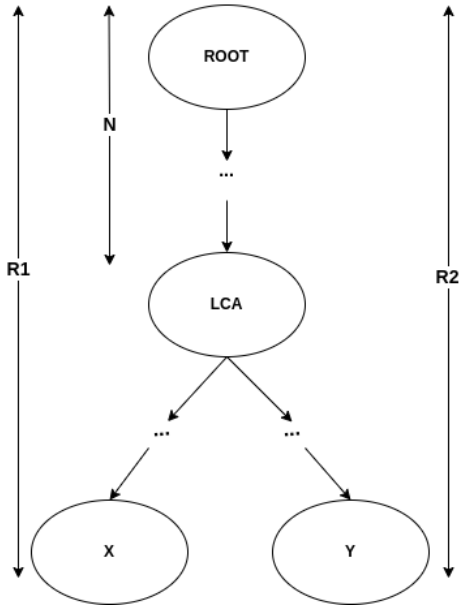


FIGURE 2.7: Example of a concept hierarchy for *WP*

between two concepts in order to include how far two concepts are semantically. *Extended Wu Palmer Similarity (XWP)*, as we term it, is defined as:

$$XWP(X,Y) = \frac{2N \cdot e^{\frac{-L\lambda}{D}}}{R_1 + R_2} \tag{2.10}$$

where L is the shortest path between concepts X, Y as follows, i.e. for every edge passed in the vertical direction a weight of 1 is given and when the direction is changed, a weight of 1 more is given. D is the depth of the whole ontology tree. R_1, R_2 is the number of arcs between concepts X, Y and the root node respectively and N is the distance of the LCA for the concepts X, Y from the root node. Finally, λ is 1 for distant neighborhood concepts (i.e., concepts from a different hierarchy) and 0 for concepts from the same hierarchy (i.e., concepts coming from the same *close* ancestor).

Definition 2.9.4. The *Jaccard index*, also known as the *Jaccard Similarity (JS)*, is a statistic used for evaluating the *similarity* or *diversity* of finite sets and it is categorized as a semantic similarity measure for sets of concepts. Formally, the *JS* of sets X and Y

can be computed as the size of intersection divided by the size of the union of two sets:

$$JS(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{2.11}$$

- $|X \cap Y|$: gives the number of members shared between both sets.
- $|X \cup Y|$: gives the total number of members in both sets (shared and unshared).

The Jaccard Similarity will be 0 if the two sets do not share any values and 1 if the two sets are identical.

2.10 Related Work

To begin with, a recommender algorithm along with a social interactions mechanism are equipped in a tourist guide application, presented by [96], with the goal of locating undiscovered touristic POIs. An integrated Collaborative Filtering system is employed to suggest touristic locations that have been already rated by the users. The authors of [86] proposed a picture-based recommender technique for proposing tourism sites to a specific individual. Particularly, any set of images are picked by the user and then is imported into computer vision models that generate a profile with respect to the tourist's interests. Sarkar et al. [83] introduced a Crow Search Optimization-based Hybrid Recommendation model capable of generating precise recommendations to travelers though the combination of Content-based and Collaborative Filtering techniques. In their algorithm, undiscovered items are presented to a user based on similar item selection from past interactions. Thus, the similarity between the items is calculated via a combined employment of Jaccard Similarity and Simple Matching Coefficient (SMC) as similarity metrics. In addition, their method is improved as Collaborative Filtering for Java (CF4J) is enhanced with the Jaccard Similarity metric. They assess experimentally their approach through data provided by the well-known travel-related platform, TripAdvisor. Another approach of developing a Hybrid recommender algorithm for tourism that combines Content-based, Collaborative and Demographic filtering techniques proposed by Kbaier et al. [42]. Riyani et al. [75] introduced a hybrid Recommender System that works under implicit ratings and semantic similarity in order to provide effective recommendations in a different domain. In greater detail, their suggested method is divided into three filtering components: content-based, collaborative, and hybrid; and it makes use of tagging attributes to provide more relevant suggestions on discussion groups. The WordNet lexical database [26] is used to extract the semantic importance of the tags, which are then grouped in a hierarchical framework depending on their semantic relevance.

In a nutshell, Content-based RSs exploit information from previous user-system interactions to provide effective recommendations. Particularly, these systems suggest items that are very similar to items the user has liked in previous interactions. Collaborative Filtering algorithms, by contrast, analyze user's ratings to calculate the similarity between them. CF recommenders intuitively work under the assumption that, when items are rated by two users in a similar way, they probably share same interests and will provide similar ratings to other items. As mentioned in the introduction, the *cold-start* problem refers to the inability of such algorithms to make recommendations, which are personalized and relevant to users' preferences, due to the lack of ratings, when dealing with new users or items-to-recommend [27]; this

problem is of vital importance in applications related to the tourism domain [100], and thus several ways to tackle cold-start in this domain are proposed in the literature. For instance, Feng et al. [27] propose a ranking model which is a hybrid between a CF ratings-oriented and a Bayesian personalized pairwise ranking-oriented one; while Zheng et al. [100] employ a hybrid CF-based method that refines item opinion reputation and user preferences, by utilizing opinion-mining technology to mine text reviews and subsequently assess the destinations' preference ranking when matched against preferred features chosen by users via means of an artificial interaction module.

Now, Bayesian recommenders explicitly model their underlying uncertainty regarding user preferences by efficiently maintaining and exploiting prior distributions over their user models, with the purpose of progressively improving the accuracy of their recommendations. Specifically, Bayesian methods has been proved quite significant in tackling uncertainty in applications with implicit feedback. In [67] researchers derived a generic optimization criterion using a Bayesian analysis of the problem and presented a learning algorithm which is able to provide solutions which satisfy the aforementioned criterion. Sun et al. [94] demonstrated a method using Bayesian Graph Convolutional Neural Networks for modeling the interaction of users with items in implicit recommendations setting. Nevertheless, Bayesian approaches are of course able to deal with and improve recommendations where users share explicit feedback to the system. Zheng and Koren [99] proposed a Bayesian hierarchical model to generate personalized recommendations. In more detail, their model is relied on the idea that the system may effectively learn a new user's preferences as the number of users contained in the system is large and by exploiting their preference-related data via the hierarchical model.

For instance, concerning the domain of our interest, the authors of [22] focus on travel personalized recommendations and demonstrate interesting applications by utilizing freely available community-contributed photos. In more detail, they introduce a probabilistic Bayesian framework for mobile recommendations and test it on over ten million images gathered from 19 large cities. PersTour [48] is a recommendation algorithm which suggests POIs to tourists by analyzing the tourist's preferences and a specific POI's popularity. Another Recommender System for suggesting POIs to visitors was introduced by [56]. In their technique the most appropriate POIs for a visitor are recommended as Sentiment Intensity Analysis (SIA) is employed to analyze the reviews provided for all the available POIs. In addition, PersTour is capable of modifying the duration a visitor should spend for a POI with respect to her preferences. Furthermore, Babas et al. [4] propose a Bayesian approach that models *both* the items under recommendation and the user preferences by the same underlying distribution. (In their case, this distribution was the multivariate Gaussian distribution.) This is what they term as the *"You Are What You Consume"* concept. Their recommendation movie technique exhibits a performance results that are comparable to the (at the time) state-of-the-art of a popular Collaborative Filtering method for movie recommendations, as shown via an experimental evaluating on data from the MovieLens dataset. It is interesting that this is achieved without the approach having to consult previously obtained or processed data about the user. Also noteworthy is that in [91],[92] the authors exploit the *"You Are What You Consume"* concept of [4] in order to construct Bayesian recommenders for the tourist domain. We adopt certain aspects of the approach of [4, 91, 92] during the preference elicitation process of our *Hybrid RS*.

Finally, although providing solutions to a different application domain, a work

that bears certain similarities with our, because of the fact that it utilizes semantic similarity measures and hierarchies of items-to-recommend, is that of [10]. The authors, in particular, provide a museum RS for cellphones that merges a Content-based approach with semantic similarity measures and a semantically enriched Collaborative Filtering method to propose relevant museum exhibits to the visitors. Contextual post filtering is implemented to create a personalized tour of the museum depending on the physical environment and location of the visitor. In opposition to our approach, the authors only employ *Wu-Palmer* and *Jaccard* similarity metrics; they do not utilize Bayesian inference; they limit themselves to museum collections since their method is strongly dependent on the usage of specialized ontologies for artworks and cultural heritage; while that paper comes without any kind of evaluation for the approach it proposes.

Chapter 3

Novel Semantic Similarity Measures for The Recommendation Domain

In this chapter we discuss how several semantic similarity measures can be employed to calculate the similarity between POIs instead of concepts. First, we construct a hierarchy tree that contains POIs in its leaves and apply well-known hierarchy similarity measures. Moreover, we propose three different variations of Jaccard Similarity, namely Extended Jaccard Similarity, Weighted Jaccard Similarity and Weighted Extended Jaccard Similarity, that do not require the existence of any hierarchy structure, while various examples are given to describe their functionality. Finally, we evaluate experimentally the novel similarity measures we introduce by computing the similarity between POIs of the aforementioned municipality in Crete and recommending those that "fit" on the user's interests or preferences.

3.1 Hierarchy Similarity Measures

Our method distributes the points of interests, i.e., the places or activities that a destination can provide, among the leaves of a specialized hierarchical structure. The design of the POIs' hierarchy for a tourist destination plays a significant role in our approach. All the possible POIs must be examined extensively and should be added as leaves in the hierarchy tree based on the type of the amusement they offer. For instance, a monastery belongs to the cultural POIs or a wine bar comes under the leisure POIs and more specifically under the bar node. An example of POIs' hierarchy tree is depicted in Figure 3.1 and will be used to review the several semantic similarity (or distance) measures. Let W_1 and W_2 denote two different wine bars, C_1 a classic bar and M_1 a monastery.

3.1.1 Path Length Similarity (PLS)

We calculate the *PLS* between nodes of the POIs' hierarchy tree depicted in Figure 3.1. The similarity measure will be examined between node W_1 and the other nodes:

- Path Length Similarity:
  - $PLS(W1, W2) = 1 + 1 = 2$
  - $PLS(W1, C1) = 2 + 2 = 4$
  - $PLS(W1, M1) = 6 + 4 = 10$

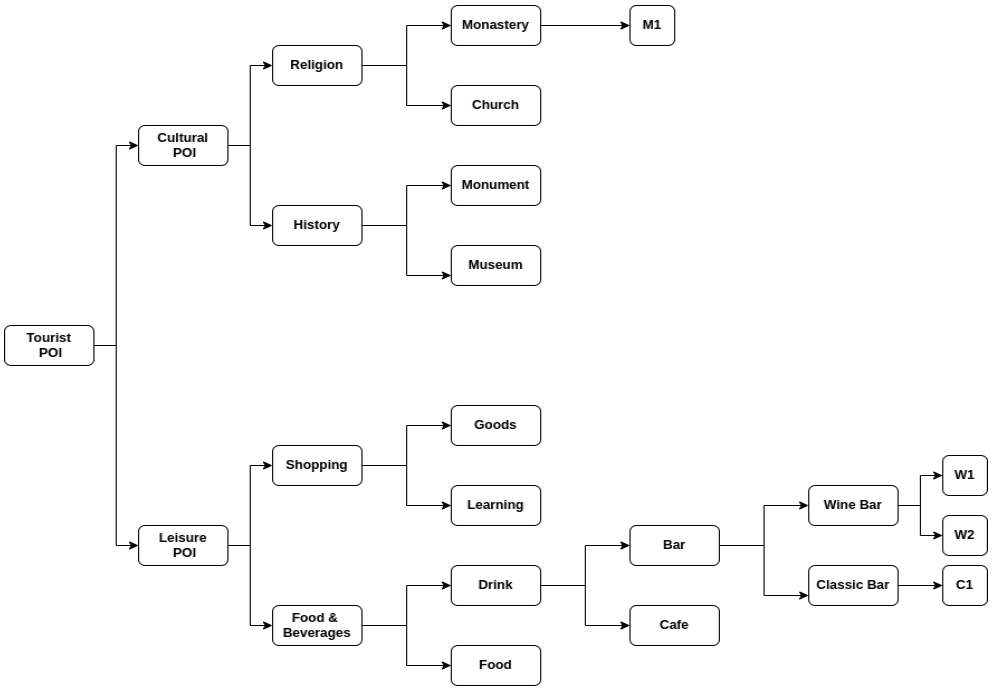


FIGURE 3.1: Example of a POIs' hierarchy tree (constructed based on real-world data)

Intuitively, the more similar two nodes are, the shortest will be the distance between them. In comparison to the other nodes in our example, W_2 is closer to W_1.

3.1.2 Wu-Palmer Similarity

An example applying *Wu-Palmer similarity* measure on the hierarchy tree of Figure 3.1 is presented, demonstrating that W_1 is more similar to W_2 than C_1 or M_1.

- Wu-Palmer similarity:
  - $WP(W1, W2) = \frac{2 \cdot 5}{6+6} = \frac{10}{12} = 0.8333$
  - $WP(W1, C1) = \frac{2 \cdot 4}{6+6} = \frac{8}{12} = 0.6666$
  - $WP(W1, M1) = \frac{2 \cdot 0}{6+4} = \frac{0}{10} = 0.0000$

3.1.3 Extended Wu-Palmer Similarity

This specific method calculates the similarity distance based on the Wu-Palmer similarity. XWP similarity, however, uses a new factor (λ) which checks if the two comparing nodes appertain to the same hierarchy or not. We display its definition equation (Eq 2.10) again here for convenience:

$$XWP(X, Y) = \frac{2N \cdot e^{\frac{-L\lambda}{D}}}{R_1 + R_2}.$$

Like WP, similar concepts have XWP close to 1 and the less similar have XWP close to 0. Only the identical nodes have XWP equal to 1. In order to understand the functionality of the metric, we are going to employ it, considering concepts of the same hierarchy depending on their distance from their LCA, denoted as $distance_{LCA}$. Again the example of Figure 3.1 will be used with two different approaches. Firstly,

we assume that concepts belong in the same hierarchy if they have the exact same (immediate) parent, i.e. their $distance_{LCA} = 1$:

- $\lambda = 0$ if $distance_{LCA} = 1$, otherwise $\lambda = 1$
  - $XWP(W1, W2) = \frac{2 \cdot 5 \cdot e^0}{6+6} = \frac{10}{12} = 0.8333$
  - $XWP(W1, C1) = \frac{2 \cdot 4 \cdot e^{\frac{-5 \cdot 1}{6}}}{6+6} = \frac{8 \cdot 0.4345}{12} = 0.2897$
  - $XWP(W1, M1) = \frac{2 \cdot 0 \cdot e^{\frac{-11 \cdot 1}{6}}}{6+4} = \frac{0}{10} = 0.0000$

Then, we consider concepts of the same hierarchy those with distance from their common parent less than or equal to 2.

- $\lambda = 0$ if $distance_{LCA} \leq 2$, otherwise $\lambda = 1$
  - $XWP(W1, W2) = \frac{2 \cdot 5 \cdot e^0}{6+6} = \frac{10}{12} = 0.8333$
  - $XWP(W1, C1) = \frac{2 \cdot 4 \cdot e^0}{6+6} = \frac{8}{12} = 0.6666$
  - $XWP(W1, M1) = \frac{2 \cdot 0 \cdot e^{\frac{-11 \cdot 1}{6}}}{6+4} = \frac{0}{10} = 0.0000$

We recognize that the choice of λ offers flexibility in the way similarity is defined. For instance, comparing the similarity measures of the two approaches for nodes W_1 and C_1, we note that when we assume that the concepts to the same hierarchy (having $\lambda = 0$), they are deemed as more similar by the measure.

3.2 Non-Hierarchy Similarity Measures

As already stated, Jaccard Similarity belongs to the category of semantic similarity measures for sets of concepts. In our case, the finite set refers to some POIs from a tourist destination. We provide an evaluation of the metric by calculating similarities between two POIs. Note that the existence of a hierarchy tree is not needed, while if such exists, we are unaware of their hierarchy tree distance. Furthermore, we present three novel metrics inspired by Jaccard Similarity and examine them as similarity measures for sets of concepts.

3.2.1 Jaccard Similarity

We now offer a concrete example to demonstrate how the Jaccard Similarity will help us measure the similarity of two nodes without having a hierarchy tree at hand. In our example we are provided with two different sets of features, A and B, each one corresponding to a tourist POI in Agios Nikolaos, Crete, with the features (members of sets) presented in Table 3.1.[1]

The size of the intersection ($|A \cap B|$) is given by the number of the underlined features, and the size of union ($|A \cup B|$) of POIs is computed by the total number of members in both sets. Thus, JS is calculated as:

- $A \cap B$ = {Cost, Wine variety, Natural landscape, Food variety, Wine cellar, Wifi, Wheelchair accessibility, Type}

[1]Some members of the sets in these examples represent synthetic data which have been added for the purpose of showcasing the behavior of the various metrics based on different input data.

Feature	*Set A (POI A)*	*Set B (POI B)*
Type:	Wine bar	Monastery
Cost:	70 euro	80 euro
Wine Variety:	{Agiorgitiko,...,Thrapsathiri}	{Liatiko,...,Thrapsathiri}
Natural landscape:	Sea	Mountain
Food variety:	{Fingerfood, Restaurant Menu}	Olive oil tasting
Wine cellar:	Available for tour	Available for tour
Wheelchair accessibility:	Wheelchair entrance	Wheelchair entrance
Wifi:	Available for customers	Available for customers
Serving Type:	Serves dine in	-
Secondary Type:	Restaurant	-
Live music:	Available	-
Other drinks:	{Cocktails, Spirits}	-
Wine production:	-	Making its own local wine
Cultural info:	-	Important cultural heritage...
Wine store:	-	Wine available for take away
Wine distribution:	-	Available
Type of tour:	-	Organized tour

TABLE 3.1: Example of features and feature values of sets A,B

- $A \cup B$ = {Cost, Wine variety, Natural landscape, Food variety, Wine cellar, Other drinks, Wifi, Wine production, Cultural info, Wine store, Wheelchair Accessibility, Type, Secondary Type, Wine distribution, Type of tour, Serving Type, Live music}
- $JS(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{8}{17} = 0.4705$

The result of the above calculation indicates the low similarity of the two finite sets (*A*,*B*). Despite the fact that sets have many shared members, the available information of the POIs' features allows us to infer their dissimilarity.

Let us now give a second example, using the Jaccard Similarity for the above sets with fewer members of their union. The number of their shared members (intersection) is going to be identical with the number of the first example. The estimation of JS of the new sets C, D is presented below:

- $C \cap D$ = {Cost, Wine variety, Natural landscape, Food variety, Wine cellar, Wifi, Wheelchair accessibility, Type}
- $C \cup D$ = {Cost, Wine variety, Natural landscape, Food variety, Wine cellar, Other drinks, Wifi, Wine production, Cultural info, Wheelchair accessibility, Type, Wine distribution, Type of tour}
- $JS(C,D) = \frac{|C \cap D|}{|C \cup D|} = \frac{8}{13} = 0.6153$

The idea of this example is to demonstrate how the similarity of the sets *C, D* is affected by the number of features in the union. We see that the Jaccard Similarity is greater when their union members are fewer than the first example (*A, B*), and the shared members remained untouched. In this case, the *JS* equals to 0.6153, which compared to 0.4705, constitutes a significant increase in the estimate of similarity of the sets. However, the "Set C" POI is still a wine bar, while the "Set D" POI is still a monastery, thus the Jaccard Similarity now provides us with a "signal" regarding semantic similarity that is rather distorted or flawed.

Another computation of *JS* will be presented by removing more members of the sets of the first example. Specifically, we remove more members of the union, some

intersection members of the two sets and then we calculate their JS. The new sets E, F are:

- $E \cap F$ = {Cost, Wine variety, Food variety, Wifi, Wheelchair Accessibility, Type}
- $E \cup F$ = {Cost, Wine variety, Food variety, Other drinks, Wifi, Wine production, Wheelchair Accessibility, Type}
- $JS(E,F) = \frac{|E \cap F|}{|E \cup F|} = \frac{6}{8} = 0.75$

The JS of the third example is greater than the outcomes of the first two, indicating that the compared sets are more similar and JS is very sensitive to the number of features of the sets.

3.2.2 Extended Jaccard Similarity

The examples above indicated that Jaccard Similarity is sensitive to the number of items in the comparing sets. Thus, it can be problematic when intending to employ JS. As such, we put forward an extended version of the JS, one that is potentially more appropriate for similarity comparisons. *Extended Jaccard Similarity (EJS)* implements a comparison between the shared members of two sets and uses only the members with equal values for the calculation of the Jaccard index. Formally, we define the Extended Jaccard Similarity as:

Definition 3.2.1. Extended Jaccard Similarity

$$EJS(X,Y) = \frac{\sum_{i \in X \cap Y} x_i}{|X \cup Y|} \tag{3.1}$$

$$\text{where,} \quad x_i = \begin{cases} 1, & \text{if} \quad v_i^X = v_i^Y \\ 0, & \text{otherwise.} \end{cases} \quad \text{with} \quad i \in X \cap Y \tag{3.2}$$

v_i^X and v_i^Y denote the values of a given feature i for each set X and Y respectively, while feature i is a member of the intersection of the sets that are being compared.

Now we can test the functionality of our proposed metric, using the examples presented above. For each example, we are going to exploit the values of the shared members, seen in Table 3.2, in order to calculate the extended version of Jaccard Similarity.

Sets A and B	*Sets C and D*	*Sets E and F*
$NaturalLand_A \neq NaturalLand_B$	$NaturalLand_C \neq NaturalLand_D$	$FoodVariety_E \neq FoodVariety_F$
$FoodVariety_A \neq FoodVariety_B$	$FoodVariety_C \neq FoodVariety_D$	$Cost_E = Cost_F$
$WineCellar_A = WineCellar_B$	$WineCellar_C = WineCellar_D$	$WineVariety_E = WineVariety_F$
$Cost_A = Cost_B$	$Cost_C = Cost_D$	$Wifi_E = Wifi_F$
$WineVariety_A = WineVariety_B$	$WineVariety_C = WineVariety_D$	$Type_E \neq Type_F$
$Wifi_A = Wifi_B$	$Wifi_C = Wifi_D$	$Wheelchair_E = Wheelchair_F$
$Type_A \neq Type_B$	$Type_C \neq Type_D$	
$Wheelchair_A = Wheelchair_B$	$Wheelchair_C = Wheelchair_D$	

TABLE 3.2: Feature values comparison of the example sets

As we noticed, five of eight shared members of the given sets are exactly equal, in the sense that they are the same and have exactly the same values[2]. The extended Jaccard similarity is computed as:

- $EJS(A,B) = \frac{\sum_{i \in A \cap B} x_i}{|A \cup B|} = \frac{5}{17} = 0.2941$

The outcome of the extended version of Jaccard Similarity represents the difference when each one of the shared members for both sets A, B is being checked. The values of the shared members provide more information about the features each POI has. For instance, a better perspective of the similarity of the sets is acquired when an intersection member such as "Type" is compared.

We attempt to deploy our new measure for sets C, D. Firstly, the members of their intersection are being checked in Table 3.2. Then, the Extended Jaccard Similarity is calculated:

- $EJS(C,D) = \frac{\sum_{i \in C \cap D} x_i}{|C \cup D|} = \frac{5}{13} = 0.3846$

The similarity of sets C, D is greater than the similarity of sets A, B, because of the reduced number of union members of the second case. Less information about the POIs' features provides a higher similarity result, however the EJS is capable of still capturing the dissimilarity between the two POIs: their similarity score is still low, as even though they share several features, these do not have the same values. The Extended Jaccard Similarity can produce better results than the original version, giving us the opportunity to recognize the similarity distance between sets of a hierarchy tree.

We are going to deploy our measure for sets E, F, and compare the results with the original version of the Jaccard index. The Extended Jaccard Similarity is computed as:

- $EJS(E,F) = \frac{\sum_{i \in E \cap F} x_i}{|E \cup F|} = \frac{4}{8} = 0.50$

It turns out that two very similar sets (such as E, F) based on the Jaccard index, become less similar by employing our proposed similarity measure. The comparison of the feature values provides more informative outcomes about the similarity distance between two sets, and is less sensitive than the original version of Jaccard similarity to the reduction of the features of the sets.

3.2.3 Weighted Jaccard Similarity

Earlier we presented the deployment of Jaccard index for evaluating the similarity (or diversity) of finite sets and specifically for POIs through several examples. The outcomes showed that Jaccard index can be quite efficient when we have been provided with enough information about the features of the compared sets. The Extended Jaccard Similarity, on the other hand, is capable of creating reliable results by including the important content given by the feature values of the compared POIs: the dissimilarity in feature values led to POIs being deemed less similar by EJS than when employing the original JS version. However, it was observed that this reduction in similarity by EJS can be rather drastic, while EJS does not take user preferences into account.

[2]We suppose the cost values are equal if the difference in cost is not very large. In addition, the wine varieties of the two POIs are equal, because the wine variety of set B is a subset of wine variety of set A.

We thus now come up with a new technique for discovering the similarity of two finite sets, which unlike the previous methods, utilizes weights for all POIs features based on the user's preferences. We assume these weights sum up to 1.

Formally, we define the *Weighted Jaccard similarity (WJS)*[3] for two finite sets (X, Y) as:

Definition 3.2.2. Weighted Jaccard Similarity

$$WJS(X,Y) = \frac{\sum_{i \in X \cap Y} w_i}{\sum_{j \in X \cup Y} w_j} \tag{3.3}$$

where, w_i is the weight of i^{th} member of the intersection of two compared sets and w_j is the weight of j^{th} member of the union. The Weighted Jaccard Similarity will be 0 if the two sets do not share any values and 1 if the two sets are identical. We now demonstrate the effectiveness of the Weighted Jaccard Similarity by employing it on the sets E and F described in the previous examples. Our example makes apparent the importance of having features of the intersection with high weight values.

We provide four different example applications of the weighted index by presenting four different assignments of weight values of the E, F sets described above and we are going to calculate the WJS for each one. For the purpose of distinguishing the features that are members of the intersection we underline them in our examples. Specifically, the scenarios are presented in Table 3.3.

Weights	*Scenario 1*	*Scenario 2*	*Scenario 3*	*Scenario 4*
$\underline{w_{Cost}}$:	0.125	0.05	0.30	0.20
$\underline{w_{WineVariety}}$:	0.125	0.40	0.02	0.40
$\underline{w_{FoodVariety}}$:	0.125	0.20	0.02	0.025
$\underline{w_{Wifi}}$:	0.125	0.05	0.02	0.025
$\underline{w_{Wheelchair}}$:	0.125	0.05	0.02	0.025
$\underline{w_{Type}}$:	0.125	0.05	0.02	0.025
$w_{OtherDrinks}$:	0.125	0.10	0.30	0.15
$w_{WineProd}$:	0.125	0.10	0.30	0.15

TABLE 3.3: Weight assignment for each scenario

The weight utilization of *scenario 1* describes that the user does not have a special bias for the features presented in Table 3.3. In occasions where the weight values of "wine variety" are high such as *scenarios 2, 4* present the significance of including a "wine variety" as feature by the POIs based the user's preferences.

The Weighted Jaccard Similarity for each scenario is calculated based on Equation 3.3 for sets E, F in the first line of Table 3.4. Observing the examples, while we change the weight values the outcomes of the Weighted Jaccard Similarity vary. In cases where the weight values of the intersection members are high the compared sets are very similar like scenarios 2,4. In contrast, when features, that are only union members, have higher weight values the similarity of the sets decreased unambiguously (i.e., scenario 3).

[3]Note that the *Ruzicka similarity* [79] is sometimes mentioned as *weighted Jaccard similarity* in the literature; while the Tanimoto coefficient [77] is also called *extended Jaccard* . Those terms should not be confused with the concepts defined in our work.

Measures	*Scenario 1*	*Scenario 2*	*Scenario 3*	*Scenario 4*
WJS:	0.75	0.80	0.40	0.70
WEJS:	0.50	0.55	0.36	0.65

TABLE 3.4: WJS and WEJS values per scenario (for the E, F sets)

3.2.4 Weighted Extended Jaccard Similarity

Up to this point, we defined two new metrics based on the Jaccard Similarity and each one is capable of measuring the similarity of two sets. EJS metric extracts the valuable information about the similarity of two POIs through the comparison of the feature values. On the other hand, WJS employs a weight factor that reflects the importance of each value based on the user's interests and exploits it to calculate the similarity of the sets. It would be very interesting to develop a new metric as the combination of the aforementioned measures for the calculation of the similarity of the sets.

Weighted Extended Jaccard Similarity (WEJS), as we could name it, takes advantage of the weight values (inspired by Weighted Jaccard Similarity) of the different features when only their feature values are equal (inspired by Extended Jaccard Similarity) in order to measure the similarity. Content of the POIs provided by the feature values and the user's preferences are taken into consideration in order to create a new metric capable of including all the important information and measuring the similarity between sets. Specifically, we assume all the members (or features) that belong to possible compared sets are assigned to weights and all these weights sum up to 1. The proposed metric implements a comparison between the shared members of two sets and uses the weights of the members with equal values. Formally, the $WEJS$ is defined as:

Definition 3.2.3. Weighted Extended Jaccard similarity

$$WEJS(X,Y) = \frac{\sum_{i \in X \cap Y} \alpha \cdot w_i}{\sum_{j \in X \cup Y} w_j} \tag{3.4}$$

$$\alpha = \begin{cases} 1, & \text{if} \quad v_i^X = v_i^Y \\ 0, & \text{otherwise.} \end{cases} \quad \text{with} \quad i \in X \cap Y \tag{3.5}$$

where, v_i^X and v_i^Y denote the values of a given feature i for each set X and Y respectively, while feature i is a member of the intersection of the sets. w_i is the weight of i^{th} member of the intersection of two compared sets and w_j is the weight of j^{th} member of the union. α is the variable that checks when the compared values of the features are equal. The $WEJS$ will be 0 if the two sets do not share any values and 1 if the two sets are identical. We demonstrate this index's effectiveness by employing it on the sets E, F for the scenarios of Table 3.3 and comparing their feature values in Table 3.2. Like the WJS the union members are the only features that we assigned weights, so the sum of their weights is equal to 1. The results of each different scenario are depicted in Table 3.4.

In the first case, where the weight values of all features are equal, the result of our new metric is similar to the outcome of Extended Jaccard Similarity for sets E, F. However, these compared sets are less similar than the corresponding example in Weighted Jaccard Similarity because of the unequal values of features "Type" and "Food Variety". We should recall that, the comparison implemented between a wine

bar and a monastery and the information provided by their feature values (such as "Type") is significant for the qualitative improvement of the similarity results. Furthermore, the specific metric tries to point out the importance of having equal feature values of the intersection members, specially when their weight values are high (scenarios 2, 4). For example, in scenarios 3 and 4 of the above example the similarity of the compared sets changes dramatically, as soon as we assign a higher weight to the "Cost" feature which has the similar value for both sets.

3.3 Experimental Evaluation

In this section, we evaluate the metrics introduced earlier on a real world dataset including 430 POIs in the vicinity of Agios Nikolaos, Crete, Greece. As we already mentioned, the creation of the dataset is one of our contributions in this work, and uses 58 features to describe all the included POIs. The first 14 features represent the *character* of a POI (e.g., how cultural or how family friendly a POI is) and take values in the range of $[1, 10]$, with such a value corresponding to the degree that this feature describes a POI.[4] The *characteristics*, such as the "Type" of a POI and the amenities a POI can provide, are captured by the last 44 features. An example of these features is presented in Table 3.1.

3.3.1 Hierarchy Similarity Measures Experiments

We construct a hierarchy, which has a similar form as the tree structure in Figure 3.1, by exploiting the 58 features that describe all the available POIs. We note that each POI may belong to more than one node of the hierarchy tree (i.e., having multiple ancestors) depending on the information provided by the 58 features. Two POIs are chosen randomly as inputs for the purpose of examining the distance measures of Section 3.1. In a real world application, *input POIs* represent items that a user "likes" in order to get recommendations similar to her choices, while the recommendation technique is based on hierarchy similarity measures.

The similarity distance results between the first input (*Kritsa's Gorge*, *ancestor = gorge*), the second input (*Minos*, *ancestor = Men Casual*) and the remaining POIs of the dataset, are illustrated in Table 3.5. The 20 most similar POIs are provided as a ranking list alongside their names, ancestor nodes and similarity outcomes based on *PLS, WP similarity and two different versions of XWP similarity* (XWP^1,XWP^2). We remind the reader that lower score values denote higher similarity objects in the case of *PLS*; while larger values denote higher similarity objects in cases on *WP similarity* and *XWP similarity*. Furthermore, *WP*, XWP^1 and XWP^2 similarity outcomes are depicted together, since they return the same *ranking* of recommendations (which is expected since they operate similarly). As described earlier (see Section 2.9) the *WP*, XWP^1 and XWP^2 metrics capture the similarities among items by exploiting the whole hierarchy tree structure, as they consider *(i)* the distance from the hierarchy root of the compared POIs; and *(ii)* the distance of the compared POIs' LCA from the hierarchy root. XWP^1 considers nodes as originating from the same hierarchy and as such having a $\lambda = 0$ when their $distance_{LCA} = 1$, and sets $\lambda = 1$ for $distance_{LCA} > 1$; while XWP^2 sets λ to 1 for those items with $distance_{LCA} > 2$ (otherwise $\lambda = 0$). Still, XWP^1 does rank the POIs in a meaningful way (albeit with a lower score than *WP* or XWP^2).

[4]Note that those features also correspond to preference-related data collected from actual tourists via questionnaires.

Input POI: Kritsa's Gorge	*Ancestor: Gorge*		*Input POI: Kritsa's Gorge*	*Ancestor: Gorge*			
Recommended items (ranked)	***Ancestor***	***PLS***	***Recommended items (ranked)***	***Ancestor***	***WP***	***XWP^1***	***XWP^2***
Alsos Ag. Xaralampous	Grove	4	Akra Vangi	Cape	0.7142	0.4788	0.7142
Kritsa	Mountain Village	4	Tou Listi o Spilios	Hill	0.7142	0.4788	0.7142
Loutsi	Hill	4	Thilakas	Mountaintop	0.7142	0.4788	0.7142
Oxia	Mountaintop	4	Oxia	Mountaintop	0.7142	0.4788	0.7142
Thilakas	Mountaintop	4	Loutsi	Hill	0.7142	0.4788	0.7142
Tou Listi o Spilios	Hill	4	Kritsa	Mountain Village	0.7142	0.4788	0.7142
Akra Vangi	Cape	4	Alsos Ag. Xaralampous	Grove	0.7142	0.4788	0.7142
Elounda Square	Square	6	Plaz EOT Agios Nikolaos	Park	0.5714	0.3136	0.3136
Kitroplatia square	Square	6	Kitroplatia square	Square	0.5714	0.3136	0.3136
Plaz EOT Agios Nikolaos	Park	6	Elounda Square	Square	0.5714	0.3136	0.3136
Agii Pantes	Island	7	Mirabello Bay	Bay	0.4615	0.2291	0.2291
Almiros	River	7	Mikri gefyra	Bridge	0.4615	0.2291	0.2291
Elounda harbor	Harbor	7	Marina	Marina	0.4615	0.2291	0.2291
Kalydon	Island	7	Lake Voulismeni	Lake	0.4615	0.2291	0.2291
Kanali Elountas	Canal	7	Kanali Elountas	Canal	0.4615	0.2291	0.2291
Lake Voulismeni	Lake	7	Kalydon	Island	0.4615	0.2291	0.2291
Marina	Marina	7	Elounda harbor	Harbor	0.4615	0.2291	0.2291
Mikri gefyra	Bridge	7	Almiros	River	0.4615	0.2291	0.2291
Mirabello Bay	Bay	7	Agioi Pantes	Island	0.4615	0.2291	0.2291
Agioi Pantes Beach	Non Organized Beach	8	cave Thief	Non Organized Beach	0.4285	0.1925	0.1925
Input POI: Minos	***Ancestor: Men Casual***		***Input POI: Minos***	***Ancestor: Men Casual***			
Recommended items (ranked)	***Ancestor***	***PLS***	***Recommended items (ranked)***	***Ancestor***	***WP***	***XWP^1***	***XWP^2***
Asia market	Men Casual	2	Elxi	Men Casual	0.9	0.9	0.9
Blanc du Nil	Men Casual	2	sakt boutique	Men Casual	0.9	0.9	0.9
Chinese shop	Men Casual	2	Soho	Men Casual	0.9	0.9	0.9
Cosmos Fashion Store	Men Casual	2	Roz Mari	Men Casual	0.9	0.9	0.9
CretaCotton	Men Casual	2	Riviera Woman - Man	Men Casual	0.9	0.9	0.9
Gusto Store	Men Casual	2	Petit Fashion Store	Men Casual	0.9	0.9	0.9
Mano a mano	Men Casual	2	New fashion	Men Casual	0.9	0.9	0.9
New fashion	Men Casual	2	Mano a mano	Men Casual	0.9	0.9	0.9
Petit Fashion Store	Men Casual	2	Gusto Store	Men Casual	0.9	0.9	0.9
Riviera Woman - Man	Men Casual	2	CretaCotton	Men Casual	0.9	0.9	0.9
Roz Mari	Men Casual	2	Cosmos Fashion Store	Men Casual	0.9	0.9	0.9
Soho	Men Casual	2	Chinese shop	Men Casual	0.9	0.9	0.9
sakt boutique	Men Casual	2	Blanc du Nil	Men Casual	0.9	0.9	0.9
Elxi	Men Casual	2	Asia market	Men Casual	0.9	0.9	0.9
Alexisport	Men Sports	4	Superdry	Men Sports	0.8	0.5362	0.8
Cosmos Sport	Men Sports	4	MINOS Skate,Surf shop	Men Sports	0.8	0.5362	0.8
Direct Adventure	Men Sports	4	Kourinos Sports	Men Sports	0.8	0.5362	0.8
Kourinos Sports	Men Sports	4	Direct Adventure	Men Sports	0.8	0.5362	0.8
MINOS Skate,Surf shop	Men Sports	4	Cosmos Sport	Men Sports	0.8	0.5362	0.8
Superdry	Men Sports	4	Alexisport	Men Sports	0.8	0.5362	0.8

TABLE 3.5: hierarchy similarity measures results

In Table 3.5 the ranking lists for both inputs are very similar for all aforementioned metrics. However the WP, XWP^1 and XWP^2 metrics provide more informative scores regarding to the similarity between two compared POIs. For instance, *Alexisport* (i.e., the output POI for input POI *Minos*) has $PLS = 4$; while *Oxia* (i.e., the output POI for input POI *Kritsa's Gorge*) has also $PLS = 4$. As such, in both scenarios the *PLS* scores are identical (equal to 4), signifying simply that the output POIs lie in the same distance from their input POIs, as calculated with respect to their *LCA* for each input POI respectively. Nevertheless, more informative similarity scores appear when WP, XWP^1 and XWP^2 are employed as our hierarchy tree is an asymmetrical structure (i.e., trees have varying numbers of children per node, and different records might lie at different depths). For the examples above, *Alexisport* has $WP = 0.8$ while *Oxia* has $WP = 0.7142$, as the depth of the compared POIs is taken into account by the WP metric.

Notice that, naturally, for any hierarchy-based measure to function properly, it is crucial to have a well-defined hierarchy tree in place. However, given the potentially enormous number of POIs to place in a hierarchy appropriately, the use of hierarchy similarity measures requires post-filtering techniques for excluding a portion of the results. This can be done based on different factors, such as the POIs' ratings or the user's preferences.

3.3.2 Non-Hierarchy Similarity Measures Experiments

The evaluation of Jaccard Similarity *(JS)* and its proposed variants is conducted via a new series of experiments with the assistance of *Kritsa's Gorge* and three more POIs of the dataset as inputs. We remind the reader, larger similarity scores for *JS* and its variations denote higher similarity objects. In the manner of a RS application, input POIs represent items that a user "likes" and the recommendation technique is based on non-hierarchy similarity measures. While no hierarchy structure is employed and the ancestors are not provided, all the experiments present POIs (inputs and outputs) along with the "Type" feature which is part of the last 44 features of the dataset. We stress that the calculation of the *JS* and its variations for the members of each compared set, i.e., POI, is based on a subset of the last 44 features. Specifically, we consider only the features that have a non-empty field, i.e, are assigned to a value.

Table 3.6 depicts the experiments conducted by employing *JS* and Extended Jaccard Similarity *(EJS)* for each input POI respectively. The top ten recommendations are presented in accordance with the similarity score (*JS* or *EJS*), output POI's name and "Type" feature we mentioned earlier. In the case of *Moni Toplou* ("Type" = monastery), *JS* outputs low similarity scores and none of the top ten recommendations includes another item of the same "Type" feature, because of the incapability to integrate information about the feature values. On the other hand, *EJS* does incorporate such data, and thus the ranking changes to better reflect POIs similarities or differences. For instance, the similarity scores for the input POI (*Moni Toplou*)-related items are lower for *EJS*, because the compared feature values of the input are not equal to the corresponding feature values of the other POIs. However, a POI with equal "Type" feature (monastery) now appeared in the first place, while *JS* had ranked a "monument" at the top and had failed to output a "monastery" as a "top-ten" similar item. In the other three cases, *JS* outcomes have large similarity scores in the top ten places of the ranking list, because they share many members (i.e., intersection members), but it is problematic for inputs such as *Skala fusion taverna* ("Type" = restaurant) where a portion of the most similar items have "Type" = fast food. *EJS* similarity scores are also lower than the respective *JS* outcomes because of the comparison the POIs' features values for the remaining cases. We should note that *JS* and *EJS* create only *non-personalized* recommendations as no information about user's preferences is included. However, as is evident in the results, *EJS* results are more fine-grained, with outputs' scores belonging in a wider range of values; while *JS* scores are characterized by much homogeneity. The similarity scores for both metrics are sensitive to the number of features given by the compared POIs (i.e., features assigned to some value) and are large specially when enough information is provided such as the cases of *Fysalida* and *Skala fusion taverna* that are taken as inputs.

Weighted Jaccard Similarity *(WJS)* and Weighted Extended Jaccard Similarity *(WEJS)* are examined via a series of experiments where user's preferences are taken into consideration. Two synthetic users, whose preferences are captured by 14 features, identical with the first 14 features of our dataset, are depicted in the first two rows of Table 3.7. The respective 14 features of the 4 input POIs presented above, are also given in Table 3.7. Notably, the two users have similar feature vectors with two of the four given POIs. As such, *User0* will be referred as a *monastery-user*, since she is a user with a feature vector (i.e., the 14 features provided in Table 3.7) similar

Input POI: Moni Toplou	*Type:Monastery*		*Input POI: Moni Toplou*	*Type:Monastery*	
JS	***Recommended items (ranked)***	***Type***	***EJS***	***Recommended items (ranked)***	***Type***
0.4166	Windmills Poros Elounda	Monument	0.2857	I.M. Panagia Faneromenis	Monastery
0.4166	Saint Athanasios	Church	0.2307	I.N. Agias Paraskevis	Church
0.4166	I.N. Agiou Nektariou	Church	0.2307	Church of Agios Georgios	Church
0.4166	I.N. Mixail Arxaggelou	Church	0.2142	Saint Athanasios	Church
0.4166	I.N. No Name	Church	0.2142	Kritsa	Attraction
0.4166	I.N. No Name	Church	0.2	Granini AE	Store
0.4166	Church of Agia Paraskevi	Church	0.1875	Murodati	Dessert
0.4166	Agios Grigorios	Church	0.1875	Filema Caffe	Cafeteria
0.3846	Siganos - Winespirits	Store	0.1764	Zygos Urban Garden	Cafeteria
0.3846	I.N. Agiou Sylla	Church	0.1764	To potiraki	Tavern
Input POI: Fysalida	*Type:Bar*		*Input POI: Fysalida*	*Type:Bar*	
JS	***Recommended items (ranked)***	***Type***	***EJS***	***Recommended items (ranked)***	***Type***
1	Allegro cocktail bar	Bar	0.5555	Manos Pool Bar Elounda	Bar
0.8571	cafe Du Lac bar	Bar	0.5555	BABEL BAR RESTAURANT	Bar
0.8571	Venue cafe Bar Zerbos Ioannis	Bar	0.5555	Allegro cocktail bar	Bar
0.8571	Swell experience	Bar	0.5	Rudis Bar	Bar
0.8571	Skipper Agios Nikolaos	Bar	0.5	GALLEY BAR	Bar
0.8571	SixtySix Shisha Cocktail Bar	Bar	0.5	Beach Bar Golden Beach	Bar
0.8571	Metzo	Bar	0.4444	Skipper	Bar
0.8571	Marios Bar	Bar	0.4444	Sirocco	Bar
0.8571	Isalos Elounda	Bar	0.4444	BEERaki Cretan Pub	Bar
0.8571	Coastal All Day Sports Bar	Bar	0.4444	Arodo Cafe Bar	Bar
Input POI: Kritsa's Gorge	*Type:Attraction*		*Input POI: Kritsa's Gorge*	*Type:Attraction*	
JS	***Recommended items (ranked)***	***Type***	***EJS***	***Recommended items (ranked)***	***Type***
1	Thilakas	Attraction	0.5	Thilakas	Attraction
1	Oikismos Dyo Prinoi	Attraction	0.5	Oikismos Dyo Prinoi	Attraction
1	Oxia	Attraction	0.5	Oxia	Attraction
1	Loutsi	Attraction	0.5	Loutsi	Attraction
1	Alsos Ag. Xaralampous	Attraction	0.5	Alsos Ag. Xaralampous	Attraction
0.75	Akra Vangi	Attraction	0.4	Akra Vangi	Attraction
0.75	Tou Listi o Spilios	Attraction	0.4	Tou Listi o Spilios	Attraction
0.75	Ancient Naxos	Attraction	0.4	Ancient Naxos	Attraction
0.6	Temple of Venus Lenika	Attraction	0.3333	Temple of Venus Lenika	Attraction
0.6	Archaeological site of Lato	Attraction	0.2857	Almiros	Attraction
Input POI: Skala fusion taverna	*Type:Restaurant*		*Input POI: Skala fusion taverna*	*Type:Restaurant*	
JS	***Recommended items (ranked)***	***Type***	***EJS***	***Recommended items (ranked)***	***Type***
0.909	BLE Katsarolakia	Restaurant	0.8	La Strada	Restaurant
0.9	Umami sushi bar	Restaurant	0.75	BLE Katsarolakia	Restaurant
0.9	TWINS	Fast Food	0.7272	Piato	Restaurant
0.9	Must-Grill House fish meat	Restaurant	0.7272	Gioma Meze	Restaurant
0.9	Al Dente	Restaurant	0.7	Sergiani Seaside	Restaurant
0.8181	evanna restaurant	Restaurant	0.6	La bouillabaisse	Restaurant
0.8181	Migomis Piano Restaurant	Restaurant	0.5384	Migomis Piano Restaurant	Restaurant
0.8	Uno	Fast Food	0.5	Arismari-Cretan Mediterranean Cuisine	Restaurant
0.8	Souvlakia Sta Orthia	Fast Food	0.4615	Must-Grill House fish meat	Restaurant
0.8	Palermo	Fast Food	0.4615	Karnagio	Restaurant

TABLE 3.6: JS, EJS Experiments

	Culture	**Sun&Sea**	**History/ Archaeology**	**Adventure/ Sports**	**Affordable Prices**	**Family Friendly Activities/Facilities**	**Rural Tourism**	**Luxury Accommodation/Leisure**	**Nightlife**	**Gastronomy/ Cuisine**	**Sea Sports**	**General Shopping**	**Shopping Local Products**	**Hiking/ Trekking**
User0	10	7	9	7	9	8	8	6	1	6	1	6	9	4
User1	4	3	2	2	7	8	2	9	6	10	1	2	1	1
Moni Toplou	9	8	9	6	10	9	9	7	1	6	1	5	9	4
Fysalida	5	4	1	1	7	8	1	8	7	10	1	1	1	1
Kritsa's Gorge	1	6	7	10	10	4	5	1	1	1	1	1	1	10
Skala fusion taverna	6	7	1	1	6	9	1	8	4	10	1	1	1	1

TABLE 3.7: Features for users and POIs

to that of *Moni Toplou* POI, which has a "Type" feature equal to "monastery"[5]. Similarly, *User1* is a *bar-user*, in the sense that she has a feature vector similar to that of *Fysalida* (whose "Type" is a "bar"). Both metrics integrate the user's preferences by assigning weights to the last 44 last features of the POIs. In order to accomplish this, cosine similarity [47] is employed between the user's feature vector and all POIs' respective feature vector (i.e., the first 14 features of the dataset). The most similar POIs are chosen, based on a threshold,[6] exported by cosine similarity, and the number of appearances of a feature (last 44 features) is counted, indicating its level of importance based on the user's preferences. We note that in this counting process, we consider only the features that have a non-empty field, i.e., they are assigned to some value. Thus, the counters are transformed into weights as we normalize them within $[0, 1]$ for all 44 features, and all sum up to 1.

[5]We remind the reader that "Type belongs to the 44 last features characterizing a POI

[6]The threshold for this series of experiments was selected to be equal to 0.7.

Table 3.8 and Table 3.9 present the *WJS* and *WEJS* similarity scores for *bar-user* and *monastery-user* respectively for the 4 input POIs. Large *WJS* values appear for both users, as no information about the feature values is imported in this similarity measure. In cases like *Fysalida*, *Kritsa's Gorge* and *Skala fusion taverna* the large *WJS* values indicate many intersection members between the input POIs and the rest of the dataset for both users. When *Moni Toplou* is given as input, the *WJS* scores are lower than the other outputs for both users, because of the reduced number of the shared members. Notably, the ranking lists for each input between the two users differ, as their preferences are taken into account through the integration of the weights of the 44 POIs features. However, the recommendations provided by the *WJS* metric are characterized by homogeneity and poor performance (see the cases where *Moni Toplou* and *Skala fusion taverna* are the input POI for both users, in which the "Type" of the outputs very seldomly matches that of the input).

On the other hand, *WEJS* combines the information about the content of the POIs and the user's preferences. The similarity scores are lower than the respective outcomes of *WJS* because of the feature values' comparison (i.e., 44 POIs' features). The ranking lists now differ based on each user's preferences for each input POI. In the case of *Moni Toplou*, low similarity scores in the top ten ranks indicate limited information about the feature values of the compared POIs for both users, however the items are listed in accordance with the user's interests: For instance, a store (*Granini A.E.*) which provides wines and spirits appears in the first place above the items with "Type = monastery" for the *bar-user*, while the monasteries are recommended first for the *monastery-user* because her feature vector is similar to the *Moni Toplou* monastery. We note that, items with "Type = store" or "Type = bar" are still presented in the top ten ranks, as many of them provide a wine list or wine tasting, like *Moni Toplou*, and this information is given by the features (i.e., POIs' content) of our dataset. As such, *WEJS* provides different similarity outcomes based on the different users, and is able to create diverse recommendations. In the remaining cases, where enough information is granted items with the same "Type" as the input POI are listed in the top ten ranks. Here the "richness" of the recommended items is also observed. For example, in the case of *Kritsa's Gorge* many items with "Type = attraction" are not gorges, such as *Oxia* and *Oikismos Duo Prinoi* (i.e., in Table 3.5 their ancestor node is "mountaintop") which are recommended by *WEJS* to both users.

3.4 Conclusion

In this chapter we experimented with several semantic similarity measures for computing the similarity between POIs in order to create Content-based recommendations in the tourist domain. Well-known similarity distance measures are examined by operating them upon *hierarchies of POIs*. Progressively, we build three novel versions of *Jaccard* similarity and we provide their advantages and disadvantages by employing them to measure the similarity between POIs. We end up with *WEJS*, a hierarchy-free measure, capable of recommending POIs by combining information based on the user's preferences and the feature values of POIs. *WEJS* is able to create "rich" recommendations by capturing similarities among POIs that are relevant to the user's preferences.

Input POI: Moni Toplou	*Type:Monastery*		*Input POI: Moni Toplou*	*Type:Monastery*	
WJS	***Recommended items (ranked)***	***Type***	***WEJS***	***Recommended items (ranked)***	***Type***
0.7342	Siganos - Wine,spirits	Store	0.4040	Granini A.E.	Store
0.7210	Granini A.E.	Store	0.3457	I.M. Panagia Faneromenis	Monastery
0.7086	Murodati	Dessert	0.3209	Monastiri Agios Ioannis THeologos	Monastery
0.6440	Vardas Bakery	Dessert	0.3045	Filema Caffe	Cafeteria
0.6440	Vardas Bakery	Dessert	0.3045	Marina Cafe	Bar
0.6440	TRADITIONAL BAKERY	Dessert	0.3045	Ammoudi Club	Bar
0.6440	Myrodati Paradosiako	Dessert	0.2981	Meltemi Seaside	Tavern
0.6440	Lasithiotikos fournos	Dessert	0.2949	Zygos Urban Garden	Cafeteria
0.6382	Yanni's Rock Bar	Bar	0.2949	Palazzo cafe bar	Cafeteria
0.6209	Must Cafe	Cafeteria	0.2910	To potiraki	Tavern
Input POI: Fysalida	*Type:Bar*		*Input POI: Fysalida*	*Type:Bar*	
WJS	***Recommended items (ranked)***	***Type***	***WEJS***	***Recommended items (ranked)***	***Type***
1	Allegro cocktail bar	Bar	0.8826	Manos Pool Bar Elounda	Bar
0.9929	cafe Du Lac bar	Bar	0.8826	BABEL BAR-RESTAURANT	Bar
0.9929	Venue cafe Bar Zerbos Ioannis	Bar	0.8108	Rudis Bar	Bar
0.9929	Swell experience	Bar	0.8108	GALLEY BAR	Bar
0.9929	Skipper Agios Nikolaos	Bar	0.8108	Beach Bar Golden Beach Voulisma	Bar
0.9929	SixtySix Shisha Cocktail Bar	Bar	0.8108	Arodo Cafe Bar	Bar
0.9929	Métzo	Bar	0.7641	Sirocco	Bar
0.9929	Marios Bar	Bar	0.7641	BEERaki Cretan Pub	Bar
0.9929	Isalos Elounda	Bar	0.7641	Alyggos Bar	Bar
0.9929	Coastal All Day Sports Bar	Bar	0.7015	Allegro cocktail bar	Bar
Input POI: Kritsa's Gorge	*Type:Attraction*		*Input POI: Kritsa's Gorge*	*Type:Attraction*	
WJS	***Recommended items (ranked)***	***Type***	***WEJS***	***Recommended items (ranked)***	***Type***
1	Thilakas	Attraction	0.9198	Thilakas	Attraction
1	Oikismos Dyo Prinoi	Attraction	0.9198	Oikismos Dyo Prinoi	Attraction
1	Oxia	Attraction	0.9198	Oxia	Attraction
1	Loutsi	Attraction	0.9198	Loutsi	Attraction
1	Alsos Ag. Xaralampous	Attraction	0.9198	Alsos Ag. Xaralampous	Attraction
0.9962	Tou Listi o Spilios	Attraction	0.9163	Tou Listi o Spilios	Attraction
0.9962	Ancient Naxos	Attraction	0.9163	Ancient Naxos	Attraction
0.9225	Akra Vangi	Attraction	0.8485	Akra Vangi	Attraction
0.9193	Temple of Venus Lenika	Attraction	0.8456	Temple of Venus Lenika	Attraction
0.9193	Archaeological site of Lato	Attraction	0.8204	Kalydon	Attraction
Input POI: Skala fusion taverna	*Type:Restaurant*		*Input POI: Skala fusion taverna*	*Type:Restaurant*	
WJS	***Recommended items (ranked)***	***Type***	***WEJS***	***Recommended items (ranked)***	***Type***
0.9943	TWINS	Fast Food	0.8943	La Strada	Restaurant
0.9943	Must-Grill House fish meat	Restaurant	0.8921	BLE Katsarolakia	Restaurant
0.9943	Al Dente	Restaurant	0.8551	Sergiani Seaside	Restaurant
0.9607	Umami sushi bar	Restaurant	0.8529	Piato	Restaurant
0.9551	Uno	Fast Food	0.8529	Gioma Meze	Restaurant
0.9551	Souvlakia Sta Orthia	Fast Food	0.7981	Must-Grill House meat	Restaurant
0.9551	Palermo	Fast Food	0.7981	Arismari-Cretan Mediterranean Cuisine	Restaurant
0.9551	Askianos	Restaurant	0.7981	Al Dente	Restaurant
0.9536	BLE Katsarolakia	Restaurant	0.7614	evanna restaurant	Restaurant
0.9483	evanna restaurant	Restaurant	0.7614	Karnagio	Restaurant

TABLE 3.8: WJS and WEJS experiments, bar-user

Input POI: Moni Toplou	*Type:Monastery*		*Input POI: Moni Toplou*	*Type:Monastery*	
WJS	***Recommended items (ranked)***	***Type***	***WEJS***	***Recommended items (ranked)***	***Type***
0.6526	Murodati	Dessert	0.4387	I.M. Panagia Faneromenis	Monastery
0.6424	Windmills Poros Elounda	Monument	0.3419	Monastiri Agios Ioannis THeologos	Monastery
0.6062	Marina Cafe	Bar	0.2436	Granini A.E.	Store
0.6062	Ammoudi Club	Bar	0.2308	Marina Cafe	Bar
0.6015	Cretan Olive Oil Farm	Tour	0.2308	Ammoudi Club	Bar
0.5920	Vardas Bakery	Dessert	0.2300	Filema Caffe	Cafeteria
0.5920	Vardas Bakery	Dessert	0.2256	Meltemi Seaside	Tavern
0.5920	TRADITIONAL BAKERY	Dessert	0.2236	Zygos Urban Garden	Cafeteria
0.5920	Myrodati Paradosiako	Dessert	0.2236	Palazzo cafe bar	Cafeteria
0.5920	Lasithiotikos fournos	Dessert	0.2217	To potiraki	Tavern
Input POI: Fysalida	***Type:Bar***		***Input POI: Fysalida***	***Type:Bar***	
WJS	***Recommended items (ranked)***	***Type***	***WEJS***	***Recommended items (ranked)***	***Type***
1	Allegro cocktail bar	Bar	0.8914	Manos Pool Bar Elounda	Bar
0.9931	cafe Du Lac bar	Bar	0.8914	BABEL BAR-RESTAURANT	Bar
0.9931	Venue cafe Bar Zerbos Ioannis	Bar	0.8029	Rudis Bar	Bar
0.9931	Swell experience	Bar	0.8029	GALLEY BAR	Bar
0.9931	Skipper Agios Nikolaos	Bar	0.8029	Beach Bar Golden Beach	Bar
0.9931	SixtySix Shisha Cocktail Bar	Bar	0.8029	Arodo Cafe Bar	Bar
0.9931	Metzo	Bar	0.7472	Sirocco	Bar
0.9931	Marios Bar	Bar	0.7472	BEERaki Cretan Pub	Bar
0.9931	Isalos Elounda	Bar	0.7472	Alyggos Bar	Bar
0.9931	Coastal All Day Sports Bar	Bar	0.7137	Allegro cocktail bar	Bar
Input POI: Kritsa's Gorge	***Type:Attraction***		***Input POI: Kritsa's Gorge***	***Type:Attraction***	
WJS	***Recommended items (ranked)***	***Type***	***WEJS***	***Recommended items (ranked)***	***Type***
1	Thilakas	Attraction	0.8965	Thilakas	Attraction
1	Oikismos Dyo Prinoi	Attraction	0.8965	Oikismos Dyo Prinoi	Attraction
1	Oxia	Attraction	0.8965	Oxia	Attraction
1	Loutsi	Attraction	0.8965	Loutsi	Attraction
1	Alsos Ag. Xaralampous	Attraction	0.8965	Alsos Ag. Xaralampous	Attraction
0.9098	Akra Vangi	Attraction	0.8156	Akra Vangi	Attraction
0.8738	Tou Listi o Spilios	Attraction	0.7834	Tou Listi o Spilios	Attraction
0.8738	Ancient Naxos	Attraction	0.7834	Ancient Naxos	Attraction
0.8686	Kalydon	Attraction	0.7761	Almiros	Attraction
0.8656	Almiros	Attraction	0.7745	Kalydon	Attraction
Input POI: Skala fusion taverna	***Type:Restaurant***		***Input POI: Skala fusion taverna***	***Type:Restaurant***	
WJS	***Recommended items (ranked)***	***Type***	***WEJS***	***Recommended items (ranked)***	***Type***
0.9944	TWINS	Fast Food	0.8913	BLE Katsarolakia	Restaurant
0.9944	Must-Grill House fish meat	Restaurant	0.8678	La Strada	Restaurant
0.9944	Al Dente	Restaurant	0.8342	Sergiani Seaside	Restaurant
0.9663	Umami sushi bar	Restaurant	0.8181	Piato	Restaurant
0.9607	Uno	Fast Food	0.8181	Gioma Meze	Restaurant
0.9607	Souvlakia Sta Orthia	Fast Food	0.7870	Must-Grill House fish meat	Restaurant
0.9607	Palermo	Fast Food	0.7870	Arismari-Cretan Mediterranean Cuisine	Restaurant
0.9607	Askianos	Restaurant	0.7870	Al Dente	Restaurant
0.9467	Taverna Stavros	Tavern	0.7534	Kouzina Restaurant	Restaurant
0.9426	BLE Katsarolakia	Restaurant	0.7534	Askianos	Restaurant

TABLE 3.9: WJS and WEJS experiments, monastery-user

Chapter 4

The Content-based Recommender

Here we describe the implementation of the Content-based (Cb) recommendation algorithm. As mentioned in the introduction, this algorithm makes use of two main sub-components: *(i)* a *hierarchy* similarity measure-based sub-component and *(ii)* a hybrid similarity measure-based component which takes as input the output of the aforementioned sub-component and employs a novel non-hierarchy similarity measure (specifically, *WEJS*, the *Weighted Extended Jaccard Similarity*). We hereby refer to the former sub-component as the *Hierarchy Similarity Measure-based Recommendations (Hierarchy SMbR)* algorithm, and to the latter as the *Hybrid Similarity Measure-based Recommendations (Hybrid SMbR)* algorithm.

The *Hierarchy SMbR* calculates the most similar POIs with respect to the user preferences via a well-established hierarchy tree structure. The *Hybrid SMbR* makes use of the aforementioned hierarchy similarity measure and the *WEJS*—which takes into account both the POIs' features (i.e., *content* of POIs) and the user preferences, as we explain in detail later in this section. Our *Cb Recommender* constitutes an algorithmic engine combining the *Hierarchy* and *Hybrid* SMbRs along with two preference elicitation-related parts: *(i)* the *Show Generic Images*; *(ii)* the *Build User Profile*, as illustrated in Figure 4.1. The *POIs Database* stores and also provides all the essential information for the proper operation of our proposed algorithmic engine. The connection of the three components of our implementation with the database is depicted with orange arrows in Figure 4.1, each of which inserts different input data to the sub-components, as required for their operation.

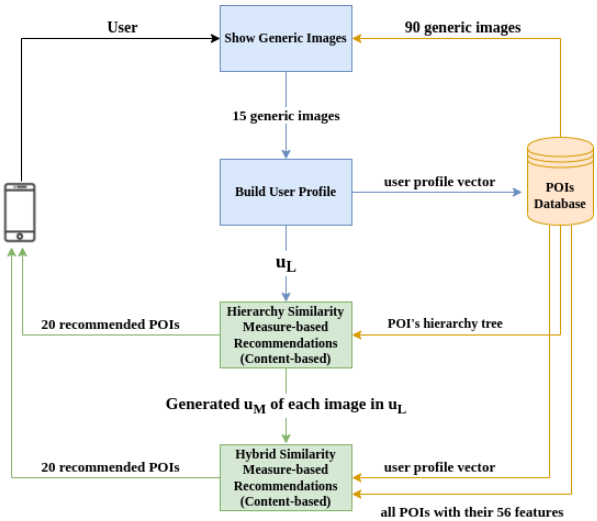


FIGURE 4.1: The algorithmic engine of the Cb Recommender

Now, the construction of the *user profile* is implemented by the combination of the *Show Generic Images* part of the engine that takes as input 90 *generic images* retrieved

from the database, and the *Build User Profile* part. In order for the *Hierarchy SMbR* algorithm to work properly, it is important to have an appropriate hierarchy of POIs. To this end, we have constructed a hierarchy of POIs for our application domain, the resort town of Agios Nikolaos, in close collaboration with local stakeholders (e.g., the municipality authorities). A snapshot of the constructed tree hierarchy, containing some of the POIs' categories we identified, is shown in Figure 4.2.[1] In our hierarchy, the POIs are inserted in the last layer of the tree structure as leaves, under a layer containing the ninety (90) different POI categories included in the hierarchy. We included 430 POIs in our hierarchy, located in Agios Nikolaos and its vicinity.

The POIs' database stores our hierarchy and, of course, POIs, and specifies 56 features (each one with a distinct value) for all 430 POIs. The POI's *character* is represented by the first 12[2] features, with values ranging from 1 to 10, reflecting the extent to which each feature characterizes the POI in question. These 12 "POI character" features were identified as such given the findings of a survey we conducted among real tourists visiting Agios Nikolaos in order to collect data on the preferences of tourists visiting the city.[3] For instance, the POI's *character* could be a combination of cultural and leisure features, e.g. a monastery that offers wine to its visitors. The other 44 features represent the *characteristics* of a POI (e.g., the amenities a POI may offer). In contrast to the *Hierarchy SMbR* which only requires the POIs hierarchy in order to function, the *Hybrid SMbR* exploits all such features of all POIs stored in our database, along with the *user profile vector*. We now proceed to describe all the main parts of the *Cb Recommender* algorithmic engine in detail.

4.1 Constructing a User Profile for Cb recommendations

The first process executed by our algorithm is the construction of the user profile. For this purpose, we use two different components, namely the *Show Generic Images* and *Build User Profile*. Ninety (90) *generic images* (or, *generic POIs*) are also stored in our database, and are each related to a specific category of POIs. In other words, these *generic POIs* are pictures that illustrate POIs which are not contained to the database; and, importantly, are represented only by the first 12 features which demonstrate the *character* of the *generic POI* depicted in them.

The hierarchy tree we utilize in our algorithm is asymmetrical, i.e. it can have an arbitrary number of children per node, and all branches may have different length. Generic POIs and actual POIs are inserted as leafs in the tree structure, lying under its layer that contains 90 nodes corresponding to the 90 different POI categories. Although every generic image, corresponding as it does to a specific category of POIs, lies on a specific leaf of the hierarchy tree, actual POIs may lie on more than one leafs. As mentioned earlier, in Figure 4.2 we visualize a snapshot of the POIs' hierarchy structure, generated by real-world data containing specific examples of POIs and categories. Namely, G1 is a gallery and C1 is a cocktail bar, connected to the categories "Gallery" and "Cocktail Bar" respectively. We should highlight that the percentage of POIs and generic images related to "Leisure" is 80%, while those related to "Culture" belong to the other 20%. Those proportions are derived by the responses of actual tourists that participated in the aforementioned survey.

[1] Our complete hierarchy tree can be found in Appendix A.

[2] The 14 features introduced in Section 3.3 were reduced to 12 as through experimental analysis we observed better performance of our algorithm.

[3] In Appendix B we include the questionnaire we compiled and used in the aforementioned survey.

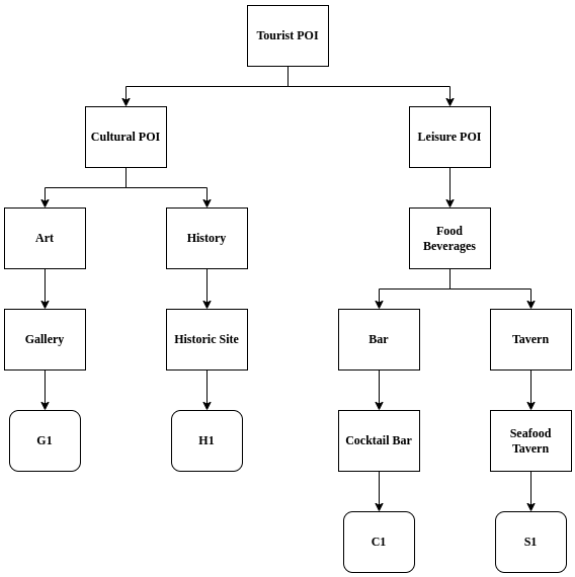


FIGURE 4.2: Part of the constructed POIs' hierarchy tree. G1, H1, C1 and S1 are leave-nodes corresponding to specific POIs

The *Show Generic Images* part of our algorithmic engine depicted in Figure 4.1 is responsible for selecting randomly and presenting to the user only 15[4] out of 90 *generic images* preserving the proportion of leisure and cultural generic POIs. Subsequently, these 15 *generic images* are imported to the *Build User Profile*. Based on [86], we adopt an elicitation process in which the user either "likes" or "dislikes" each of the aforementioned 15 *generic images* that are presented to her in sequence, resulting in the generation of the *user profile vector*. Notably, the user is allowed to terminate this process at any time (i.e, she does not have to interact with all 15 images).

To build the *user profile*, we act as follows. In accordance with the work of. [4], we represent the *user profile* as a *vector* containing the first 12 features of the POIs in our dataset—that is, the features which demonstrate the *character* of a POI. We then populate this vector with values as follows: We first construct the set of all generic images that the user has "liked"—let this set be denoted as u_L. Then, for each one of the 12 "character features", we calculate the average of its values in the u_L set of "liked" generic images, and insert this average as the value of the corresponding element in the *user profile vector*.

4.2 Hierarchy Similarity Measure-based Recommendations

The *Hierarchy SMbR* is a sub-component of the Cb recommender algorithm which is capable of providing recommendations via the hierarchy similarity measure called *XWP* introduced in Section 2.9. We remind that this metric is based on a similarity measure introduced in [89], which is an extended version of *Wu-Palmer (WP)* [98].

The extended version of *WP*, or XWP, computes the semantic relation between the ontologies X, Y as:

[4]The choice of presenting to the user "15" generic images was made since, as [23] indicates, completing a survey with 15 questions requires five to seven minutes only; while pressing a "like" or "dislike" indicator on an image is arguably much faster than responding to a question. In practice, experimentation with our real-world app shows that a real user "likes" or "dislikes" the 15 images presented to her within 45 seconds

$$XWP(X,Y) = \frac{2N \cdot e^{\frac{-L\lambda}{D}}}{R_1 + R_2} \tag{4.1}$$

where R_1, R_2 represent the number of edges between the *root node* and the ontologies X, Y respectively, while N is the number of edges between the *root node* and the *Least Common Ancestor (LCA)* of X and Y. D is the depth of the hierarchy structure and L represents the minimum distance between ontologies X and Y. Also, the λ factor is zero when the compared ontologies' LCA is in their neighborhood, otherwise equals one. In our instantiation of the *XWP* measure, we define the neighborhood of a particular node to be the set of all other nodes which have a distance less than or equal to 2 from it.

The left part of Figure 4.3 depicts an overview of the *Hierarchy SMbR*-based recommendation process. Specifically, the u_L vector of "liked" images is given as an input to the *Hierarchy SMbR* sub-component and for every "liked" generic image contained in this set we compute the *XWP* similarity distance between it and each POI in our database. Now, the M most relevant POIs to each generic image i according to *XWP* are stored in a set, denoted as u^i_M, alongside their similarity scores. For instance, as Figure 4.3 illustrates, if the user has "liked" N generic images during the elicitation process, our system will create one set for every "liked" image, i.e., $u^1_M, u^2_M, \ldots, u^N_M$. The POIs included in all u_M sets are grouped together, and their corresponding *XWP* similarity scores are transformed into a probability distribution by a normalization process—i.e., the *Normalization Process* part of the *Hierarchy SMbR* component (illustrated in blue) in Figure 4.3. Finally, if the standalone operation of this sub-module is desired, we sample out 20 POIs in order to be presented to the user.

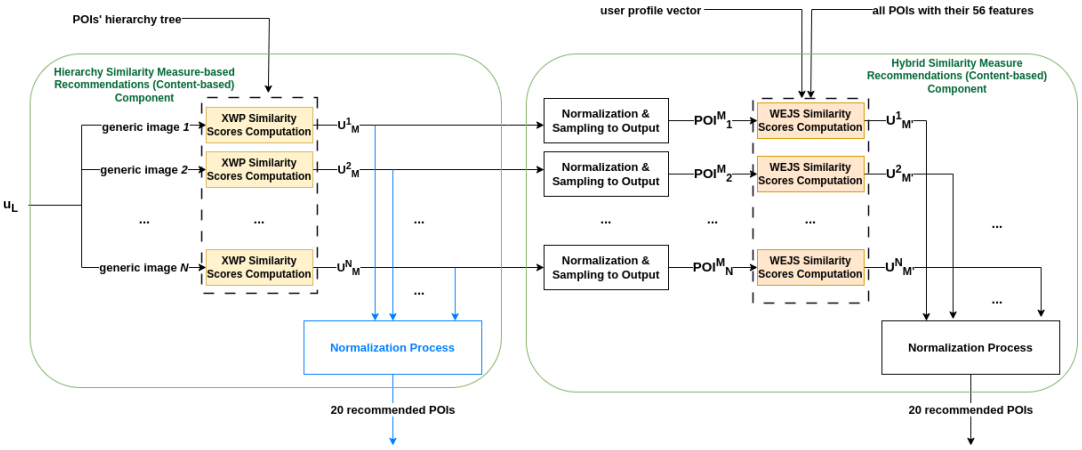


FIGURE 4.3: Overview of our Content-based (Cb) Recommender

4.3 Hybrid Similarity Measure-based Recommendations

However, the *Hierarchy SMbR* sub-component need not operate alone, but can be extended to our ("hybrid") *Cb recommender* as follows. We now add a sub-component that performs *Hybrid similarity measure-based recommendations* (right part of Figure 4.3), and which incorporates the Weighted Extended Jaccard Similarity (*WEJS*), combining it with the (hierarchy-based) *XWP*. In detail, as illustrated in Figure 4.3, the *Hybrid SMbR* sub-component takes three different inputs: *(i)* the u_M sets produced by

Hierarchy SMbR (we remind the reader that each such set corresponds to a generic image in u_L); *(ii)* the *user profile vector* (i.e., the 12 feature-based vector generated by the *Build User Profile* component); and *(iii)* the POIs included in the database alongside their features. Our system transforms the *XWP* similarity outcomes into probability distributions for each u_M produced by the hierarchy similarity measure via the *Normalization & Sampling to Output* component. Specifically, after this transformation each *XWP* similarity score of a POI in a specific u_M set is mapped to a value in the range of [0,1], while the summation of all such values for POIs in the u_M in question equals to 1. Given these probabilities, we sample out of each u_M (corresponding to an image in u_L), one POI for computing its *WEJS* similarity score (via the Eq. 4.2 formula explained below in detail). As Figure 4.3 indicates, we denote the drawn POI of u^i_M set as POI^i_M, where $i \in \{1, \cdots, N\}$. After the employment of *WEJS* similarity score, for each POI^i_M, a new set, denoted as $u^i_{M'}$, is generated, and it contains at most M' similar POIs along with their corresponding *WEJS* similarity scores. Following this, all generated sets $u^i_{M'}$, $i \in \{1, \cdots, N\}$, are concatenated, and their *WEJS* similarity scores are converted into probabilities. Finally the system samples out 20 POIs, and recommends them to the user.

Now, we remind the reader *WEJS* is computed by the following expression:

$$WEJS(X,Y) = \frac{\sum_{i \in X \cap Y} \alpha \cdot w_i}{\sum_{j \in X \cup Y} w_j}, \quad \alpha = \begin{cases} 1, & \text{if} \quad v_i^X = v_i^Y \\ 0, & \text{otherwise.} \end{cases} \qquad with \qquad i \in X \cap Y \quad (4.2)$$

where X and Y in our application domain correspond to two distinct POIs—i.e., to the sets of these POIs' feature values. Moreover, v_i^X and v_i^Y represent the value of the i^{th} feature of X's and Y's intersection; thus, α is 1 if they are identical, otherwise it is 0. w_i (respectively w_j) is given by the output of Algorithm 1, and corresponds to the weight of the i^{th} (j^{th}) feature of the intersection (union) of POIs X, Y. As such, *WEJS* measure exploits the generated weights of all POI's last 44 features (*i.e. characteristics*) with the purpose of comparing the intersection members (i.e., features) of two POIs, and it utilizes only those with equal values. Intuitively, two items would be considered by *WEJS* highly similar and also of high recommendation value to a user, if they share the exact same characteristics (i.e., share features with the exact same values), and these characteristics are deemed important (i.e., they are highly "weighted" by the user as observed in Section 3.3.2).

As mentioned, Algorithm 1 presents the method of generating the weights vector with respect to the user preferences for the POI's *characteristics*, or, as we term it, the *weights user profile vector*. As seen in Algorithm 1, the cosine similarity distance metric [47] is employed between the twelve features of the *user profile vector* and each actual POI's twelve *character* features. The most similar POIs are selected according to a *cosine threshold*[5], and the total appearances of their last 44 features with non-empty values are counted. Finally, these features' counters are normalized into weights comprising the generated $weightsProfile_u$ of the user, to be used in the *WEJS* calculation.

4.4 Experimental Evaluation of the Content-based Algorithm

In this section we assess the performance of our Cb algorithm on real-world data comprising of 430 points of interest from the city of Agios Nikolaos by using a lab computer with AMD Ryzen 7 3700x 8-core processor x 16 CPU with a GeForce RTX

[5]This threshold was empirically set to 0.70 during in our experiments.

Algorithm 1 Weights generation of the user profile

$profile_u \leftarrow$ user profile vector
$POIs \leftarrow$ all POIs with their 56 features
$counters \leftarrow$ count vector of length 44 initialized to zero
for each POI in $POIs$ **do**
    $character_{POI} \leftarrow$ first 12 features of POI
    $characteristics_{POI} \leftarrow$ last 44 features of POI
    if $cosine_{sim}(profile_u, character_{POI}) \geq 0.70$ **then**
        for each f in $characteristics_{POI}$ **do**
            if $v_f \neq \emptyset$ **then**
                $counters_f \leftarrow counters_f + 1$
$weightsProfile_u \leftarrow normalize(counters)$

3080 GPU. We collected information related to *users' preferences* from questionnaires filled out by 150 actual short-term visitors and used this data to generate a set of 600 synthetic users. We categorized the synthetic users to six different age classes (100 tourists per class), namely 17-25, 26-35, 36-45, 46-55, 56-67, and 67+.

In order to generate the synthetic users' profiles, we proceeded as follows. We first computed an average profile for each age class, containing the average values of the 12 features representing user preferences, as listed in the filled-in questionnaires of the real tourists belonging to that particular class. Specifically, these 12 features were used: *Family friendly facilities and activities, Luxury Accommodation, Affordable Prices, Culture, Shopping Local Products, Sun and Sea, Rural Tourism, Sports and Adventure, General Shopping, Nightlife, Cuisine and Gastronomy, Archaeology or History*. (We remind the reader that these 12 features correspond to the 12 "POI character" features, in accordance to the "You Are What You Consume" approach). Subsequently, for each class, we sampled out 100 random user profiles from a multivariate normal distribution, generated using the mean vector from the average "real tourist" profile mentioned above; its covariance matrix is diagonal with all its eigenvalues are equal to 2.

We evaluate the performance of our Cb algorithm via two series of experiments. In our first experimental setting each (synthetic) user interacts with the Cb recommender 30 times (each time without storing any information about the model for the next run of the system), thus the set of generic images provided to her is not always the same. By contrast, in the second experimental setting, the system maintains information about past user-system interactions, which are now 10.

For each user, we utilize k-means clustering [19] to classify all POIs to two distinct categories, specifically, those that are close to her profile, termed the *relevant POIs*, while the complementary set is that of *irrelevant POIs*. Since we are interested primarily in the intersection of POIs which are relevant to those recommended by each of our algorithms, we denote as Rec_r the average cardinality of this set of POIs over all 100 users per age group, and over 30 runs per user for the first experimental setting; and over 10 runs per user for the second experimental setting.

We evaluate the performance of our algorithms by employing the well-known *precision* and *recall* measures of recommendations' quality [36]. Notably, precision is the quotient of the number of recommended relevant POIs over the number of all (relevant and irrelevant) recommended POIs; the latter, in our case, is always 20. Recall is the quotient of the number of recommended relevant POIs over the number of all (recommended and non-recommended) relevant POIs.

Now, let us provide an intuitive interpretation of precision and recall. Recall decreases when the non-recommended relevant POIs increase, while the precision is greater when the algorithm recommends fewer irrelevant POIs. Therefore, we can utilize recall in order to provide an assessment tool for users who evaluate the performance of our application based on whether they did not visit a relevant POI. On the other hand, we employ precision to model the evaluation of our algorithms from users who are interested in not visiting any irrelevant POI. Furthermore, we highlight that the number of recommended relevant POIs cannot exceed 20, however the relevant POIs are far more than this number; thus, the value of recall will always be much lower than 1. For instance, if the number of relevant POIs is 272, recall cannot exceed 0.073. Hence, recall in our results should always be assessed based on its corresponding maximum value, denoted below as *Recall Upper Bound* (or *Recall UB* for short). We focus mainly on the value of *precision* for reasons of clarity, and because we (reasonably) assume that the users' "assessment-profile" is closer to that of precision rather than recall; intuitively, short-term visitors cannot possibly explore every relevant POI with respect to their interests, but they tend to be frustrated when they spend time on irrelevant POIs.

4.4.1 First Experimental Setting

We first conduct a series of experiments in order to evaluate the performance of the semantic similarity-based algorithms that are contained in the Cb component of our system—i.e., the methods introduced in Section 4.2 and Section 4.3) of this work. Since our experiments involve synthetic users, and our Cb recommendation process requires the identification of images a synthetic user "likes". This is done by comparing the *cosine similarity* among the values of the (twelve) features of the *"actual"* (synthetic) user profile and the values of the corresponding features representing the (fifteen) "generic" images displayed to the user by the elicitation component (termed "Show Generic Images" in Fig. 4.1) of the Cb recommendation process. In case the cosine similarity between an image and the profile exceeds a threshold, set to 0.80, the image is considered to be "selected" (or "liked") by the user; otherwise it is not. The "liked" images make up the u_L set of images fed into the *Hierarchy SMbR*; while the *Build User Profile* sub-component (cf. Figure 4.1) also constructs a *user profile* that the (hybrid) Cb recommender can use for its recommendations. We now proceed with the comparison between the performance of the *hierarchy SMbR* algorithm (i.e. the output of the *Normalization Process* in blue color in Fig. 4.3) and the *hybrid SMbR* (i.e. the output of the *Normalization Process* in black color in Fig. 4.3).

In Table 4.1 and Table 4.2 we illustrate the aforementioned measures of quality (i.e. precision and recall) of our application, together with the number of relevant POIs, Rec_r and *Recall UB* for each age group. The results of Table 4.1 concern the *Hierarchy* SMbR, while those of Table 4.2 correspond to the *Hybrid* SMbR of the Cb recommender. We observe that the results (i.e., the values of *Precision*, and the respective values of *Recall* compared to *Recall UB*) between the two recommendation processes are quite satisfactory, as *Precision* is regularly higher than or close to 80%, while *Recall* reaches values close to or higher than 80% of the respective *Recall UB* ones. In general, the two *Cb* methods appear to be comparable to each other: *Hierarchy SMbR* achieves a better performance for younger users, while Hybrid SMbR attains greater values of precision and recall for the age groups 56-67 and 67+.

In addition, Table 4.3 illustrates the average precision of the algorithms across all age groups. We can observe that the *Hybrid SMbR* has an average precision value

Hierarchy Similarity Measure-based Recommendations

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.782	0.057	0.073	272.4	15.6
26-35	0.817	0.059	0.072	275.3	16.2
36-45	0.781	0.057	0.072	274.2	15.6
46-55	0.821	0.060	0.074	269.3	16.4
56-67	0.862	0.063	0.073	272.9	17.2
67+	0.684	0.055	0.080	248.9	13.7

TABLE 4.1: *Hierarchy* SMbR evaluation results

Hybrid Similarity Measure-based Recommendations

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.762	0.055	0.073	272.5	15.2
26-35	0.800	0.059	0.072	275.4	16.0
36-45	0.761	0.056	0.072	274.3	15.3
46-55	0.819	0.059	0.074	269.3	16.3
56-67	0.879	0.064	0.073	273.0	17.5
67+	0.721	0.058	0.080	249.2	14.4

TABLE 4.2: *Hybrid* SMbR evaluation results

***Hierarchy* SMbR**	***Hybrid* SMbR**
0.7911	0.7913

TABLE 4.3: Average precision values for the Cb recommendation methods

that is slightly greater than the one of the *Hierarchy SMbR*. Of course, this small difference in precision alone would not be enough in order to pick the *Hybrid SMbR* as the preferable Cb recommender system between the two. However, another drawback of the *Hierarchy SMbR* is that the u_M set is always the same for all users who "liked" a particular generic image, as such the generated recommendations would be quite similar for these users. Moreover, the method does not take into account a user profile. These characteristics clearly do not contribute to diverse or personalized recommendations.

On the other hand, the *WEJS* measure used by the *Hybrid SMbR* module combines information about the user's preferences and the content of the POIs, by taking into account the *user profile vector* and all 56 features of the POIS. As such, we expect *Hybrid SMbR* to result to recommendations that are both more personalized and more diverse. We explicate the latter point and the meaning of the term "diversity", by providing an example: when the system recommends a monastery which produces its own wine, it is probable to also recommend a liqueur store that sells the monastery's wine, even though liqueur stores are not in the vicinity of monasteries in the hierarchy tree, thus *Hierarchy SMbR* could not have possibly captured this connection.[6] Morover, the *Hybrid SMbR* is also able to exploit information about the user profile captured via the elicitation process. For those reasons, we pick the *Hybrid SMbR* to be the *Cb recommender* method of choice for our overall *Hybrid RS*, which we introduce in Chapter 5.

[6]This point is also supported by the experiments in Section 3.3 evaluating the *WEJS* measure.

4.4.2 Second Experimental Setting: Refining the User Profile over Time

Regarding the second experimental series, we assume that the system stores information regarding past user-system interactions. The *constructed* user profile, that was built in the previous interactions, is exploited in the selection of the 15 generic images to present to the user in this (current) iteration — thus effectively trying to "refine" the previously built user profile. Specifically, we employ cosine similarity between the user profile constructed in the previous iteration and the 90 generic images, for each synthetic user, in order to provide as input to the *Build User Profile* component the 15 generic images that best match this constructed-so-far user profile. Note, however, that in order to preserve an element of exploration, we always ensure that 20% of the 15 images belong to cultural POIs[7] (even if this means that we have to replace some leisure POIs with "least preferred" cultural ones). In addition, the number of the "selected" (or "liked") generic images is obtained via the same process presented in the first experimental setting. The most similar (i.e, in terms of cosine similarity using a threshold set to 0.80) of 15 generic images are derived which afterwards are exploited in the construction of the u_L set and the user profile.

Hierarchy Similarity Measure-based Recommendations

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.819	0.059	0.073	272.0	16.4
26-35	0.855	0.062	0.072	275.4	17.1
36-45	0.834	0.060	0.073	273.9	16.7
46-55	0.808	0.058	0.074	269.7	16.1
56-67	0.824	0.059	0.073	273.5	16.5
67+	0.694	0.052	0.080	248.3	13.9

TABLE 4.4: *Hierarchy* SMbR evaluation results using the refining method for the user profile

Hybrid Similarity Measure-based Recommendations

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.886	0.064	0.073	271.9	17.8
26-35	0.925	0.067	0.072	275.2	18.5
36-45	0.896	0.065	0.073	274.0	17.9
46-55	0.892	0.064	0.074	270.8	17.9
56-67	0.915	0.066	0.073	273.9	18.3
67+	0.751	0.056	0.080	249.0	15.0

TABLE 4.5: *Hybrid* SMbR evaluation results using the refining method for the user profile

***Hierarchy* SMbR**	***Hybrid* SMbR**
0.805	0.877

TABLE 4.6: Average precision values for the Cb recommendation methods using the refining method for the user profile

[7] The 20% of the POIs in our real-world dataset are cultural, and that this number also corresponds to preference-related data collected by real tourists via questionnaires (i.e., ≈20% of the tourists care more about cultural-related POIs than leisure ones).

In Tables 4.4 and 4.5 we depict the precision and recall outcomes of our Cb recommender, along with the number of relevant POIs, Rec_r and *Recall UB* for each age group. The results of Table 4.4 correspond to the *Hierarchy SMbR* sub-component, while Table 4.5 illustrates the *Hybrid SMbR* component's performance. In more detail, the *Hierarchy SMbR* achieves *Precision* higher than or close to 81% and *Recall* close to or higher than 80%, expect from the age group of 67+. Now, *Hybrid SMbR* illustrates better performance in this experimental setting. Thus, for the first five groups the precision values are over 89% and the recall values are over 87%. We also observe that the results of the oldest age group of our dataset are significantly improved simply comparing them with the respective findings of the "hierarchy" recommendation process.

Moreover, Table 4.6 presents the average precision of the recommendation algorithms across all age groups (we follow the same procedure as in the first experimental setting in Table 4.3. The *Hybrid SMbR* sub-component achieves a better performance regarding to the average precision value. Consequently, we come up with the same observation as we did in the first experimental series: the *Hybrid SMbR* is more appropriate to be tested as the Cb recommender of the overall *Hybrid RS* introduced in the following chapter.

4.5 Conclusions

In this chapter we describe a the development of a Content-based Recommender System for the tourist domain. Particularly, the algorithm makes use of two main sub-components: *(i)* a hierarchy similarity measure-based sub-component and *(ii)* a hybrid similarity measure-based component which takes as input the output of the aforementioned sub-component and employs a novel non-hierarchy similarity measure (specifically, *WEJS*, *the Weighted Extended Jaccard Similarity*). We hereby refer to the former sub-component as the *Hierarchy* Similarity Measure-based Recommendations (*Hierarchy* SMbR) algorithm, and to the latter as the *Hybrid* Similarity Measure-based Recommendations (*Hybrid* SMbR) algorithm, that in the experimental evaluation provides better recommendation outcomes. The experimental assessment of the proposed Cb recommender is performed on a real-world dataset, constructed during the development of a real-world travel planning mobile application; and incorporated knowledge derived from a survey involving actual tourists. We provide two experimental setting with the purpose of testing the quality of recommendations of the two different Content-based sub-components. The results of the first setting displayed the algorithms where comparable in terms of precision. While the second experimental setting illustrated that incorporating the refining method for the user profile, the precision results of the algorithms (and more specific in the case of the *Hybrid* SMbR) can be improved significantly.

Chapter 5

A Hybrid Recommender System

Here, we describe our *Hybrid RS* which combines the aforementioned *Cb Recommender* (i.e., introduced in Chapter 4) with a *Bayesian Inference* component used for user profile elicitation. During that process, we adopt the *You Are What You Consume* approach of [4]: that is, we model the the POIs of the travel destination (i.e., items) and the users, using multivariate normal distributions over ranges of values as a common representation, describing the degree that each feature describes a specific user or item. Figure 5.1 depicts the overall architecture of the proposed system. Notice how the elicitation process of Section 4.1 (or, the blue boxes in Figure 4.1) is now replaced with a process of *Bayesian Inference*, the functionality of which we explicate immediately below.

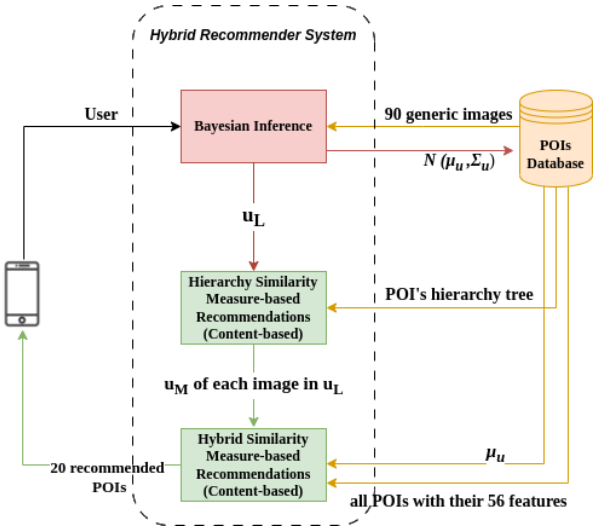


FIGURE 5.1: The algorithmic engine of the proposed Hybrid RS

5.1 Bayesian Inference of the User Profile

Initially, when a user enters our *Hybrid RS* for the first time, the preference elicitation process—i.e., the *Bayesian Inference* component in Figure 5.1)—initiates. We chose to adopt an iterative picture-based elicitation process that presents to the user a set of n generic images, each of which illustrates a specific type of POI, in order to derive information regarding her interests. Subsequently, the user selects the image which is most relevant to her interests and provides a rating for each image on a 10-level Likert scale, where 10 represents the situation where the image matches perfectly with her preferences.[1] After each iteration, our system exploits the provided rating

[1]Note that a detailed description of this process can be found in [91].

of the selected generic image and draws a specific number of samples[2] from its distribution. Intuitively, the user's rating and the logistic function [24] are utilized with the aim of computing the precise number of samples

In more detail, a high rating provided by the user for a generic image represents high similarity between the distributions of the image and the user, while a small rating means that this image does not match completely with her preferences. As such, more samples should be drawn in the case of the high rating resulting to the construction of a better model based on the user's interests. Subsequently, once our algorithm has computed the exact number of samples that should be drawn, it performs a Bayesian Inference procedure [4, 91] in order to update its belief regarding the interest's of the user. In particular, we chose to employ the Normal-Inverse Wishart (NIW) distribution which is a multivariate four-parameter family of continuous probability distributions, while it retains the property of *conjugacy* of a multivariate Gaussian distribution where the mean and covariance matrix are unknown. Generally, a closed-form for the computation of the posterior distribution is provided by the usage of conjugate priors, resulting in a computationally efficient Bayesian updating procedure. Formally, the update procedure of the prior hyperparameters—κ_0 , μ_0 (the mean vector), v_0 (degrees of freedom), and Ψ_0 (the precision matrix)— is performed by drowning samples directly from the observed data to get the posterior ones, as follows:

$$\mu_n = \frac{\kappa_0}{\kappa_0 + n} \cdot \mu_0 + \frac{n}{\kappa_0 + n} \cdot \bar{x} \tag{5.1}$$

$$\kappa_n = \kappa_0 + n \tag{5.2}$$

$$v_n = v_0 + n \tag{5.3}$$

$$\Psi_n = \Psi_0 + S + \frac{\kappa_0 \cdot n}{\kappa_0 + n} \cdot (\bar{x} - \mu_0) \cdot (\bar{x} - \mu_0)^T \tag{5.4}$$

$$S = \sum_{i=1}^{n} (x_i - \bar{x}) \cdot (x_i - \bar{x})^T \tag{5.5}$$

where n is the number of samples, x_i are the drawn samples provided by the data, $\bar{x}$ represents the sample mean and S is the scatter matrix.

$$\Sigma \sim IW(\Psi_n, v_n) \tag{5.6}$$

$$\mu \mid \Sigma \sim N(\mu_n, \Sigma/\kappa_n) \tag{5.7}$$

Thus, our algorithm can generate a normal distribution representing the user preferences at any given time: after m iterations our system has constructed a multivariate normal distribution that represents the preferences of user u, denoted as $\mathcal{N}(\mu_u, \Sigma_u)$, where μ_u and Σ_u are the mean vector and the covariance matrix of the generated distribution respectively.

Additionally, our system stores the selected image from each iteration in order to exploit such information in the *Cb recommendations* stage of the hybrid approach. As such, during the elicitation process, we create a set, denoted as u_L, which contains the images that the user selected in each iteration.

[2]The number of samples ranges form 40 to 500 depending on the user's rating.

Thus, the output of our Bayesian elicitation process is: *(i)* a multivariate Gaussian distribution, $\mathcal{N}(\mu_u, \Sigma_u)$, that describes the preferences of user u, where μ_u is employed so as to compute the weights in *WEJS* (i.e., μ_u replaces the *user profile vector* in Algorithm 1); and *(ii)* the set u_L that contains the generic images that u has selected as most preferred during the elicitation process.

5.2 Content-based Recommendations

Subsequently, our proposed system proceeds with our *Content-based (Cb) recommendation* technique. In particular, our system employs the *hybrid* similarity measure (*Hybrid SMbR*) discussed earlier, in order to produce the final set of personalized recommendations. We remind the reader, that via this hybrid measure, our algorithmic engine combines *XWP* similarity, (i.e., the hierarchy similarity measure employed in the *Hierarchy SMbR* sub-component in Figure 5.1—see Section 4.2), and our *Weighted Extended Jaccard Similarity (WEJS)* index (i.e., a non-hierarchy similarity measure employed in the *Hybrid* SMbR component in Figure 5.1—see Section 4.3), to generate recommendations that best match the user preferences.

Figure 5.2 depicts the *Cb recommender component* of the *Hybrid RS*. Notice that this figure is almost identical to Figure 4.3, with two changes: First, the *Hierarchy SMbR* sub-component is now not allowed to produce final recommendations (the blue *Normalization Process* and its output are now removed), but only provides input to the *Hybrid SMbR* sub-component. This is since our experiments in Sections 4.4.1 and 4.4.2 point to the *Hybrid SMbR* being a better choice than *Hierarchy SMbR* for a *Cb recommender* component of our final *Hybrid RS*. The second change is that μ_u replaces the *user profile vector* as input in the *WEJS* metric, since our (hybrid) *Cb recommender* is now combined with the Bayesian elicitation module that produces a user profile in the form of a $\mathcal{N}(\mu_u, \Sigma_u)$ multivariate Gaussian distribution.

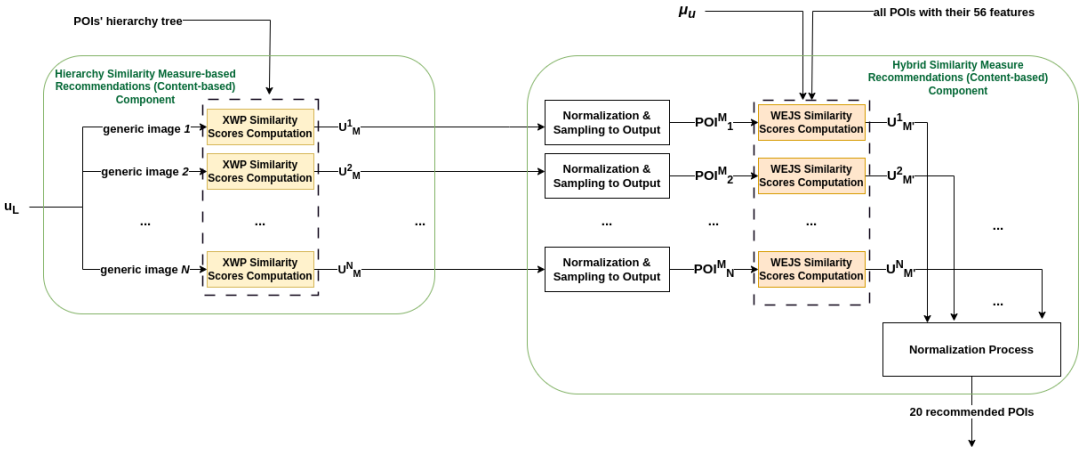


FIGURE 5.2: Overview of the Cb recommender part of the Hybrid RS

The implementation of the overall *Hybrid recommendations process* is summarized in Algorithm (2) below. The algorithm begins with the *Bayesian Inference* process, which produces the set u_L and the distribution $\mathcal{N}(\mu_u, \Sigma_u)$ describing the derived user profile. It proceeds with the generation of u_M^X sets by employing *XWP* similarity between the u_L set and the POIs in our dataset (see line 7 in Algorithm 2). Subsequently, one POI is sampled out from each u_M set via the *Normalization & Sampling to Output* component, and is provided as input to the *WEJS* measure along with the μ_u (see line 12 in Algorithm 2). Finally, for each selected POI a (new) $u_{M'}^S$ set is

created, and all such sets are concatenated (see line 13 in Algorithm 2) in order to derive the final recommendations.

Algorithm 2 Hybrid Recommender Algorithm

1: $u_L, \mathcal{N}(\mu_u, \Sigma_u) \leftarrow$ Bayesian Inference
2: $POIs \leftarrow$ all POIs with their 56 features
3: **for** each *generic image* in u_L **do**
4: $X \leftarrow$ generic image
5: **for** each *POI* in POIs **do**
6: $Y \leftarrow$ POI
7: $u_M^X \leftarrow u_M^X \cup is\_XWP\_similar?\,(X, Y,$ POIs' Hierarchy$)$
8: *Single* POI $\leftarrow$ Normalization_and_Sampling_to_Output (u_M^X)
9: $S \leftarrow$ *Single* POI
10: **for** each *POI* in POIs **do**
11: $Y \leftarrow$ POI
12: $u_{M'}^S \leftarrow u_{M'}^S \cup is\_WEJS\_similar?\,(S, Y, \mu_u)$
13: Recommendations $\leftarrow$ Recommendations $\cup\, u_{M'}^S$
14: 20 Recommended POIs $\leftarrow$ Normalization (Recommendations)

5.3 Experimental Evaluation of the Hybrid RS

In this section we evaluate the complete *Hybrid* recommender algorithm working with the same number of POIs (i.e., 430 POIS) we introduced in the assessment of the Cb algorithm in Section 4.4 by employing the well-known *k-means* clustering method. We also remind the reader our experimental dataset of users is made up of 600 synthetic users that we categorized them to six different classes — 100 synthetic tourists per age group. In addition, in our experimental setting each synthetic user interacts with the recommender 30 times, where no information is stored about the model for the next run of the system. As such, we intent to compare the evaluation outcomes of Section 4.4.1 with the performance of our proposed *Hybrid RS*.

As explained earlier, we now adopt the iterative elicitation process introduced in Section 5.1, which results into a multivariate Gaussian distribution, $\mathcal{N}(\mu_u, \Sigma_u)$, that describes the preferences of user u. In detail, we run a series of user-system interactions with varying "slates". A slate in our case corresponds to an implicit "query" presented to the user, comprising of a number n of images for the user to select a preferred one from. We experiment with varying numbers of slates $m = \{4, 5, 6\}$ posed to the user, and with varying numbers $n = \{4, 5, 6\}$ of generic images presented to the user in a slate ("query"). The testing of various such m, n combinations of *elicitation slates, images shown per slate* pairs, aims to assess the trade-off between the information offered to the algorithms (expected, naturally, to improve their performance), and the frustration caused to users when the time required from them to spend "marking" (generic) images for elicitation purposes increases [46]. Thus, as also explained in Section 5.1, during each $\langle m, n \rangle$[3] elicitation instance, the user selects the generic images that are more appealing to her, given her interests. The "selected"

[3]The same experimental settings provided in this chapter are utilized in Appendix E in order to evaluate the performance of the Bayesian RS introduced by [92]. The same $\langle m, n \rangle$ combinations are tested with the Bayesian recommender.

images are stored in the u_L set, and used alongside μ_u to recommend POIs to the user (as explained in Fig. 5.2 and Alg. 2).

In Tables 5.1 - 5.9 we present the average values of precision and recall, alongside with the (average) number of relevant POIs, Rec_r (the average number of relevant POIs among the 20 recommended) and *Recall UB* for all age groups, each of which comprises of 100 users who enter the system 30 times. Furthermore, we illustrate the average precision values for every $\langle m, n \rangle$ combination in Table 5.10.

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.833	0.060	0.073	272.4	16.6
26-35	0.869	0.063	0.072	275.3	17.3
36-45	0.849	0.062	0.073	273.7	16.9
46-55	0.844	0.062	0.074	270.1	16.8
56-67	0.840	0.061	0.073	272.6	16.8
67+	0.705	0.056	0.080	249.4	14.1

TABLE 5.1: Hybrid RS performance; ($m = 4, n = 4$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.862	0.063	0.073	272.6	17.2
26-35	0.886	0.064	0.072	275.4	17.7
36-45	0.868	0.063	0.072	274.0	17.3
46-55	0.855	0.062	0.074	269.6	17.1
56-67	0.858	0.062	0.073	272.8	17.1
67+	0.706	0.056	0.080	249.6	14.1

TABLE 5.2: Hybrid RS performance; ($m = 5, n = 5$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.853	0.062	0.073	272.7	17.0
26-35	0.886	0.064	0.072	275.2	17.7
36-45	0.869	0.063	0.072	274.3	17.3
46-55	0.861	0.063	0.074	269.6	17.2
56-67	0.854	0.062	0.073	272.8	17.0
67+	0.714	0.056	0.080	249.7	14.2

TABLE 5.3: Hybrid RS performance; ($m = 6, n = 4$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.858	0.062	0.073	272.7	17.1
26-35	0.893	0.064	0.072	275.5	17.8
36-45	0.864	0.063	0.073	273.9	17.2
46-55	0.868	0.063	0.073	270.5	17.3
56-67	0.866	0.063	0.073	272.8	17.3
67+	0.725	0.058	0.080	249.5	14.5

TABLE 5.4: Hybrid RS performance; $(m = 4, n = 6)$

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.851	0.062	0.073	273.0	17.0
26-35	0.880	0.063	0.072	275.5	17.6
36-45	0.859	0.062	0.073	273.9	17.1
46-55	0.853	0.062	0.074	269.9	17.0
56-67	0.846	0.061	0.073	273.0	16.9
67+	0.702	0.056	0.080	249.0	14.0

TABLE 5.5: Hybrid RS performance; $(m = 5, n = 4)$

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.843	0.061	0.073	272.6	16.8
26-35	0.879	0.063	0.072	275.7	17.5
36-45	0.861	0.062	0.072	274.1	17.2
46-55	0.854	0.063	0.074	269.2	17.0
56-67	0.855	0.062	0.073	272.8	17.1
67+	0.716	0.057	0.080	249.4	14.3

TABLE 5.6: Hybrid RS performance; $(m = 4, n = 5)$

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.851	0.062	0.073	272.2	17.0
26-35	0.872	0.063	0.072	275.2	17.4
36-45	0.849	0.061	0.072	274.1	16.9
46-55	0.841	0.061	0.074	270.0	16.8
56-67	0.852	0.062	0.073	272.7	17.0
67+	0.717	0.058	0.080	248.9	14.3

TABLE 5.7: Hybrid RS performance; $(m = 6, n = 5)$

As seen in Table 5.10, our *Hybrid RS* outperforms the aforementioned *Cb* algorithms (see Table 4.3) regarding the average values of *Precision* for all pairs of $\langle m, n \rangle$; and also almost always outperforms them in terms of *Recall* if we compare the results with the respective in Tables 4.1- 4.2 (the latter's values are regularly about 85% of those of *Recall UB* in Tables 5.1- 5.9, i.e., our hybrid method regularly returns about 17 recommendations that are relevant to the user profile, out of the 20 relevant recommendations that it could have possibly returned). However, slightly higher precision outcomes are observed for some age groups (namely $56-67, 67+$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.864	0.063	0.073	272.9	17.2
26-35	0.886	0.064	0.072	275.7	17.7
36-45	0.868	0.062	0.072	274.4	17.3
46-55	0.845	0.061	0.074	270.1	16.9
56-67	0.855	0.062	0.073	273.0	17.1
67+	0.708	0.059	0.080	249.0	14.6

TABLE 5.8: Hybrid RS performance; ($m = 5, n = 6$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.855	0.062	0.073	272.3	17.1
26-35	0.876	0.063	0.072	275.2	17.5
36-45	0.853	0.062	0.073	273.7	17.0
46-55	0.840	0.061	0.073	270.3	16.8
56-67	0.841	0.061	0.073	272.8	16.8
67+	0.710	0.057	0.080	249.2	14.2

TABLE 5.9: Hybrid RS performance; ($m = 6, n = 6$)

for the *Hybrid SMbR* Cb recommender (see Table 4.2). Overall, it is clear that the *Hybrid RS* has a much more stable performance than the Cb recommenders, achieving outcomes that are consistently higher than 0.84 for all pairs (apart from the 67+ age group), and consistently including about 17 POIs that are relevant to the user interests, in its 20 returned recommendations.

According to Table 5.10 our *Hybrid* recommender achieves the best results for its variant using values of $m = 4$, $n = 6$ during elicitation—i.e., when the user is presented 4 times with 6 images each time. Now, for larger values of combinations m, n, the cardinality of the set u_L is larger ($|u_L| \geq m$). This creates the possibility for more irrelevant POIs being contained in the sets u_M and $u_{M'}$, and as a result they may be presented in the final recommendations to the user— naturally, when the proportion of the irrelevant POIs is high, the precision results decrease. We can see a possible manifestation of this behaviour in the precision outcome of the $m = 6$, $n = 6$ pair, which is lower than the respective $m = 4$, $n = 6$; even though more information is potentially elicited from the user in the $m = 6$, $n = 6$ case, that information probably results to a "blurring" of the user model.

	$n = 4$	$n = 5$	$n = 6$
$m = 4$	0.823	0.834	0.846
$m = 5$	0.831	0.839	0.834
$m = 6$	0.839	0.830	0.829

TABLE 5.10: Avg. precision per $\langle m, n \rangle$ pair

Hence, given its better than all other combinations average precision listed in Table 5.10, combined with the fact that being presented with fewer slates ($m = 4$ instead of 5 or 6) results to a "lighter", less tiresome elicitation process for the user, we consider the variant of our *Hybrid RS* that uses the $\langle m = 4$ *elicitation slates*, $n = 6$ *images shown per slate*$\rangle$ pair as the one to incorporate into our final system.

	$n = 4$	$n = 5$	$n = 6$
$m = 4$	7.32	7.55	7.53
$m = 5$	7.71	7.57	7.67
$m = 6$	7.88	7.70	7.92

TABLE 5.11: Avg. execution time (in sec) per $\langle m, n \rangle$ pair

Finally, in Table 5.11 we depict the average execution time (*after the end of the elicitation process* of our *Hybrid* RS's process) for all tested combinations in Table 5.10. Here, we also report that the average recommendation time of our Content-based recommender (i.e., introduced in Chapter 4) was calculated equal to 8.82 seconds.

5.4 Conclusions

In this chapter a novel *Hybrid* Recommender System is introduced for the tourist domain, combining a Bayesian preference elicitation technique and a Content-based recommendation process introduced in Chapter 4. The experimental assessment of the proposed *Hybrid* recommender is performed on a real-world dataset, constructed during the development of a real-world travel planning mobile application; and incorporated knowledge derived from a survey involving actual tourists. We already noted that the content-based recommender component of our system, combines a hierarchy similarity measure with a novel non-hierarchical one. In addition, the first of these two methods is exclusively based on a hierarchy similarity measure (also employed in the second method) that computes the similarities between POIs only relying on their distance in the hierarchical structure that they belong. Our experimental results verify the effective performance of several variants of our proposed *Hybrid RS*, indicating that—compared to the exclusively content-based techniques in Chapter 4—they return many more POIs relevant to the inferred user profile in less execution time. In addition, when comparing the performance of the various *Hybrid RS* variants against each other, we are able to come up with a clear "winner": the most preferable *Hybrid RS* variant is the one using an $\langle m = 4$ *elicitation slates*, $n = 6$ *shown images per slate* $\rangle$ pair as input to its *Bayesian Inference* component.

Chapter 6

Conclusion & Future Work

We experimented with several semantic similarity measures for computing the similarity between POIs in order to create Content-based recommendations in the tourist domain. Well-known similarity distance measures are examined by operating them upon *hierarchies of POIs*. Progressively, we build three novel versions of *Jaccard similarity* and we provide their advantages and disadvantages by employing them to measure the similarity between POIs. We end up with *WEJS*, a hierarchy-free measure, capable of recommending POIs by combining information based on the user's preferences and the feature values of POIs. *WEJS* is able to create "rich" recommendations by capturing similarities among POIs that are relevant to the user's preferences.

Moreover, a novel Content-based Recommender System and a *Hybrid* one are introduced for the tourism domain. The latter of these two actually combines a Bayesian preference elicitation process and the aforementioned Content-based recommendation technique. The experimental assessment of the proposed *Hybrid* recommender, as well as the Content-based recommendation techniques are performed on a real-world dataset, constructed during the development of a real-world travel planning mobile application; and incorporated knowledge derived from a survey involving actual tourists. The Content-based recommender component of our system, combines a hierarchy similarity measure with a novel non-hierarchical one. Our experimental results verify the effective performance of several variants of our proposed *Hybrid RS*, indicating that—compared to the exclusively Content-based techniques—they return many more POIs relevant to the inferred user profile. In addition, when comparing the performance of the various *Hybrid RS* variants against each other, we are able to come up with a "winner": the most preferable *Hybrid RS* variant is the one using an $\langle m = 4$ *elicitation slates*, $n = 6$ *shown images per slate* $\rangle$ pair as input to its *Bayesian Inference* component.

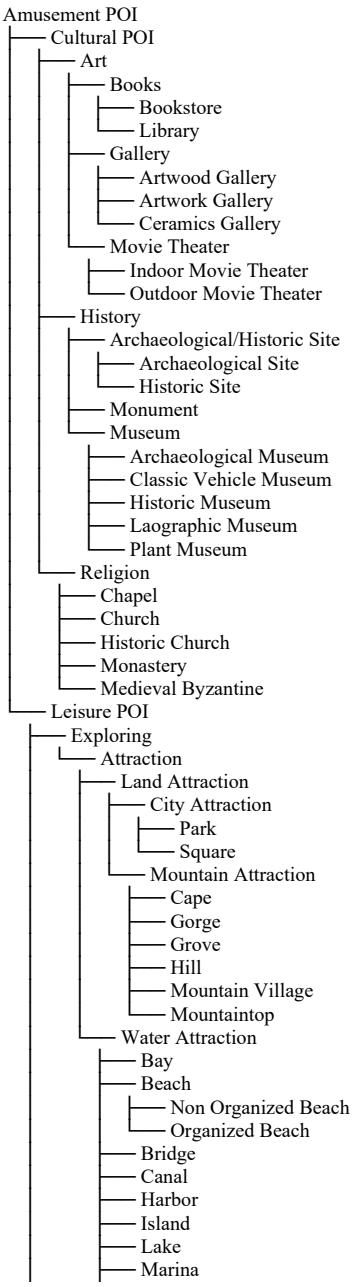
Now, despite the fact that our experimental findings are encouraging, they were performed with only synthetic users; and while we concentrated on a specific real-world dataset linked to our recommendation algorithm for the location of interest, it would be beneficial to evaluate our algorithms with various other datasets. In ongoing and future work, we aim to conduct a large-scale evaluation of our system via utilizing feedback provided by real users of our (already available online) mobile application, following all required policies and provisions for protecting users' privacy. Since different versions of our real-world application incorporate several recommender algorithms (additional to those presented in this work), such experiments will also provide us with the opportunity to compare these methods against each other, and to come up with modifications, algorithmic improvements, and even novel hybrid solutions.

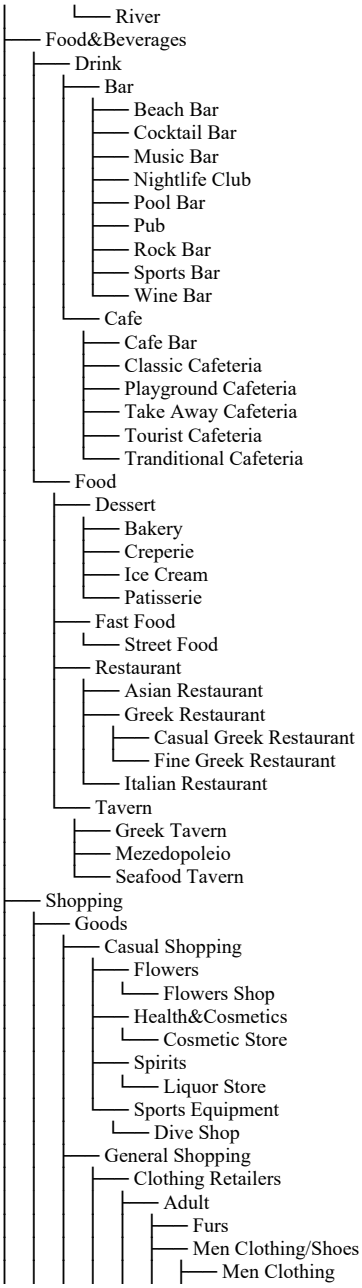
We are also eager to examine our Content-based recommendation techniques on other domains, such as the measurement of similarity among documents. Thus,

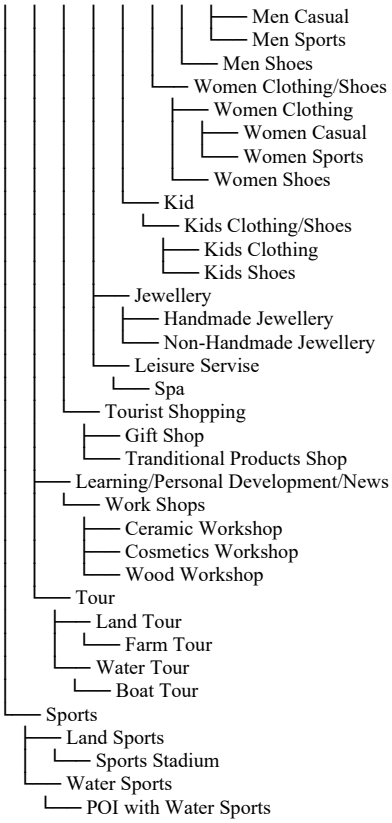
the employed metrics evaluate the semantic similarity, it would be more appropriate to test them upon the comparison of ontologies or concepts. This, in fact, does not exclude the tourism domain, where the similarities can be explored via the (text) descriptions of popular *points of interests* instead of feature values as we experimented in this work. Finally, several modifications could be applied in our proposed Content-based algorithms. For instance, the Cb recommenders could be tested with other preference elicitation processes and the weight assignment of the *WEJS* index could be replaced by a more sophisticated Machine Learning method in order to generate the weights of features based on the user preferences.

Appendix A

Hierarchy Tree




River
Food&Beverages
Drink
Bar
Beach Bar
Cocktail Bar
Music Bar
Nightlife Club
Pool Bar
Pub
Rock Bar
Sports Bar
Wine Bar
Cafe
Cafe Bar
Classic Cafeteria
Playground Cafeteria
Take Away Cafeteria
Tourist Cafeteria
Tranditional Cafeteria
Food
Dessert
Bakery
Creperie
Ice Cream
Patisserie
Fast Food
Street Food
Restaurant
Asian Restaurant
Greek Restaurant
Casual Greek Restaurant
Fine Greek Restaurant
Italian Restaurant
Tavern
Greek Tavern
Mezedopoleio
Seafood Tavern
Shopping
Goods
Casual Shopping
Flowers
Flowers Shop
Health&Cosmetics
Cosmetic Store
Spirits
Liquor Store
Sports Equipment
Dive Shop
General Shopping
Clothing Retailers
Adult
Furs
Men Clothing/Shoes
Men Clothing


Men Casual
Men Sports
Men Shoes
Women Clothing/Shoes
Women Clothing
Women Casual
Women Sports
Women Shoes
Kid
Kids Clothing/Shoes
Kids Clothing
Kids Shoes
Jewellery
Handmade Jewellery
Non-Handmade Jewellery
Leisure Servise
Spa
Tourist Shopping
Gift Shop
Tranditional Products Shop
Learning/Personal Development/News
Work Shops
Ceramic Workshop
Cosmetics Workshop
Wood Workshop
Tour
Land Tour
Farm Tour
Water Tour
Boat Tour
Sports
Land Sports
Sports Stadium
Water Sports
POI with Water Sports

Appendix B

Tourists Preferences Questionnaire

Agios Nikolaos Tourist Holiday Activities Survey

We at the Municipality of Agios Nikolaos strive to upgrade our hospitality services. Help us succeed in our mission by taking this survey. Thank you in advance.

1. Please state your nationality: ☐

2. What is your gender? Male ☐ Female ☐ Prefer not to say ☐

3. Would you like to tell us what age group you belong to?
   18-25 ☐ 26-35 ☐ 36-45 ☐ 46-55 ☐ 56 -67 ☐ 67+ ☐

4. Is Agios Nikolaos…
   …your final holiday destination, as part of an organized tour? ☐
   …a port that is part of a cruise? ☐
   …a place you chose for a daily excursion while vacationing elsewhere? ☐
   Other (Please specify): ☐

5. Are you travelling:
   Alone ☐ As a couple ☐ With friends ☐ As a family (at least one child) ☐
   With a group ☐

6. Please rate (1-5, with 5=most important) the following criteria describing your ideal vacation:

Culture	☐	Luxury accommodation and leisure	☐
Sun & Sea	☐	Nightlife	☐
History / Archaeology	☐	Gastronomy / Cuisine	☐
Adventure / Sports	☐	Sea Sports	☐
Affordable prices	☐	General shopping	☐
Family-friendly activities & facilities	☐	Shopping for local products	☐
Rural tourism	☐	Hiking / Trekking	☐

Other (Please specify): ☐

7. Duration of stay in Agios Nikolaos:
   1 day ☐ 2 days ☐ 3-5 days ☐ 6-9 days ☐ 9 days or more ☐

8. Activities that you chose/will choose during your stay in Agios Nikolaos:

Visits to archaeological sites, museums, churches	☐	Visits to beaches	☐
Visits to Restaurants, Bars, Clubs	☐	Visit to Spinalonga	☐
Daily tours	☐	Shopping (souvenirs, clothes, traditional local products, etc.)	☐
Extreme Sports	☐	Relaxing by the lake	☐

   Other (Please specify): ☐

9. Approximately how much money did you/will you spend on purchases of products and services during your stay in Agios Nikolaos (per day per person)?

   Up to €50 ☐ €51-100 ☐ €101-150 ☐ €151-200 ☐ €200 or more ☐

10. How much money did you/will you spend per category of expenses (per day per person) ?

    (A) Up to €10, **(B)** €11-€20, **(C)** €21-€50, **(D)** €51-€100, **(E)** €101+

Food & Beverages	☐	General Shopping	☐
Nightlife	☐	Local Products	☐
Tours & Driving	☐	Culture and Archaeological Sites	☐
Accommodation	☐		

Appendix C

Extra Experimental Evaluation

Here we provide some extra experimental evaluation outcomes of our Content-based algorithm introduced in Chapter 4. In more detail, the cosine threshold for acquiring the synthetic user "likes" is now set to 0.85. Tables C.1,C.2 and C.3 are illustrating the results of testing the two Cb recommenders with the same experimental setup as in Section 4.4.1.

Hierarchy Similarity Measure-based Recommendations

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.853	0.062	0.073	272.5	17.0
26-35	0.872	0.064	0.072	275.2	17.4
36-45	0.852	0.063	0.072	273.7	17.0
46-55	0.826	0.060	0.074	270.2	16.5
56-67	0.816	0.059	0.073	273.0	16.2
67+	0.623	0.046	0.080	249.2	12.4

TABLE C.1: *Hierarchy* SMbR evaluation results (cosine threshold = 0.85)

Hybrid Similarity Measure-based Recommendations

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.860	0.062	0.073	272.9	17.2
26-35	0.870	0.064	0.072	275.4	17.4
36-45	0.860	0.062	0.072	274.2	17.2
46-55	0.863	0.063	0.074	271.1	17.2
56-67	0.893	0.065	0.073	273.2	17.8
67+	0.733	0.054	0.080	249.2	14.6

TABLE C.2: *Hybrid* SMbR evaluation results (cosine threshold = 0.85)

***Hierarchy* SMbR**	***Hybrid* SMbR**
0.807	0.846

TABLE C.3: Average precision values for the Cb recommendation methods (cosine threshold = 0.85)

Tables C.4, C.5 and C.6 depict the results of the Cb algorithms (having the cosine threshold for the user "likes" equals 0.85) by utilizing the same experimental settings of Section 4.4.2.

Hierarchy Similarity Measure-based Recommendations

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.837	0.061	0.073	272.7	16.7
26-35	0.851	0.061	0.072	275.6	17.2
36-45	0.846	0.061	0.072	274.8	16.9
46-55	0.810	0.059	0.074	269.8	16.2
56-67	0.824	0.060	0.073	272.8	16.4
67+	0.673	0.050	0.080	251.5	13.4

TABLE C.4: *Hierarchy* SMbR evaluation results (cosine threshold = 0.85)

Hybrid Similarity Measure-based Recommendations

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.903	0.065	0.073	272.6	18.0
26-35	0.915	0.066	0.072	275.6	18.3
36-45	0.916	0.066	0.072	273.4	18.3
46-55	0.887	0.065	0.074	269.9	17.7
56-67	0.909	0.066	0.073	273.1	18.1
67+	0.757	0.056	0.080	250.7	15.1

TABLE C.5: *Hybrid* SMbR evaluation results (cosine threshold = 0.85)

***Hierarchy* SMbR**	***Hybrid* SMbR**
0.806	0.881

TABLE C.6: Average precision values for the Cb recommendation methods (cosine threshold = 0.85)

Furthermore, we tested our Cb algorithms with a different evaluation method in the case of the number of relevant items.

In order to measure the similarity between the user's preferences and the POIs under recommendation (and thus to be able to calculate precision) we employ the metric of cosine similarity between the (actual) user's profile and all POIs' first 12 features. Particularly, this evaluation method finds the most similar (or *relevant*) items for a given user based on a cosine threshold, and compares these via precision (we focus on the precision results only) with the ones returned by our recommender.

We note that this threshold is identical to the cosine threshold that was employed between the features of user's profile and the features of generic images in order to determine the user's "likes". Tables C.7- C.12 illustrate the results using the experimental settings of Section 4.4.1 for both cosine thresholds.

Hierarchy Similarity Measure-based Recommendations

Age Group	Precision	Relevant Items	Rec_r
17-25	0.241	73.9	4.82
26-35	0.269	80.1	5.38
36-45	0.275	90.1	5.50
46-55	0.182	51.1	3.64
56-67	0.168	45.8	3.36
67+	0.118	28.1	2.36

TABLE C.7: Cosine-based evaluation - *Hierarchy* SMbR results (cosine threshold = 0.80) - 1st experimental setting

Hybrid Similarity Measure-based Recommendations

Age Group	Precision	Relevant Items	Rec_r
17-25	0.212	0.061	73.9
26-35	0.246	0.068	80.1
36-45	0.267	0.064	90.1
46-55	0.176	0.083	51.1
56-67	0.171	0.084	45.8
67+	0.112	0.113	28.1

TABLE C.8: Cosine-based evaluation - *Hybrid* SMbR results (cosine threshold = 0.80) - 1st experimental setting

***Hierarchy* SMbR**	***Hybrid* SMbR**
0.208	0.197

TABLE C.9: Cosine-based evaluation - Average precision values for the Cb recommendation methods (cosine threshold = 0.80) - 1st experimental setting

Hierarchy Similarity Measure-based Recommendations

Age Group	Precision	Relevant Items	Rec_r
17-25	0.047	11.0	0.94
26-35	0.065	14.8	1.30
36-45	0.067	16.7	1.34
46-55	0.059	12.1	1.18
56-67	0.045	8.70	0.90
67+	0.025	4.35	0.50

TABLE C.10: Cosine-based evaluation - *Hierarchy* SMbR results (cosine threshold = 0.85) - 1st experimental setting

Hybrid Similarity Measure-based Recommendations

Age Group	Precision	Relevant Items	Rec_r
17-25	0.045	11.0	0.90
26-35	0.057	14.8	1.14
36-45	0.065	16.7	1.30
46-55	0.055	12.1	1.10
56-67	0.042	8.70	0.84
67+	0.023	4.35	0.46

TABLE C.11: Cosine-based evaluation - *Hybrid* SMbR results (cosine threshold = 0.85) - 1st experimental setting

***Hierarchy* SMbR**	***Hybrid* SMbR**
0.051	0.047

TABLE C.12: Cosine-based evaluation - Average precision values for the Cb recommendation methods (cosine threshold = 0.85)- 1st experimental setting

Finally, the outcomes regarding the experimental settings of Section 4.4.2 of the Cb algorithms using the cosine evaluation (for the two tested cosine thresholds) are presented in Tables C.13- C.18.

Hierarchy Similarity Measure-based Recommendations

Age Group	Precision	Relevant Items	Rec_r
17-25	0.228	73.9	4.56
26-35	0.264	80.1	5.28
36-45	0.294	90.1	5.88
46-55	0.200	51.1	4.00
56-67	0.172	45.8	3.44
67+	0.114	28.1	2.28

TABLE C.13: Cosine-based evaluation - *Hierarchy* SMbR results (cosine threshold = 0.80) - 2nd experimental setting

Hybrid Similarity Measure-based Recommendations

Age Group	Precision	Relevant Items	Rec_r
17-25	0.227	73.9	4.54
26-35	0.254	80.1	5.08
36-45	0.305	90.1	6.11
46-55	0.211	51.1	4.22
56-67	0.204	45.8	4.08
67+	0.124	28.1	2.48

TABLE C.14: Cosine-based evaluation - *Hybrid* SMbR results (cosine threshold = 0.80) - 2nd experimental setting

***Hierarchy* SMbR**	***Hybrid* SMbR**
0.212	0.220

TABLE C.15: Cosine-based evaluation - Average precision values for the Cb recommendation methods (cosine threshold = 0.80) - 2nd experimental setting

Hierarchy Similarity Measure-based Recommendations

Age Group	**Precision**	**Relevant Items**	Rec_r
17-25	0.063	11.0	1.26
26-35	0.068	14.8	1.36
36-45	0.077	16.7	1.54
46-55	0.064	12.1	1.28
56-67	0.048	8.70	0.96
67+	0.033	4.35	0.66

TABLE C.16: Cosine-based evaluation - *Hierarchy* SMbR results (cosine threshold = 0.85) - 2nd experimental setting

Hybrid Similarity Measure-based Recommendations

Age Group	**Precision**	**Relevant Items**	Rec_r
17-25	0.055	11.0	1.10
26-35	0.065	14.8	1.30
36-45	0.090	16.7	1.80
46-55	0.069	12.1	1.38
56-67	0.054	8.70	1.08
67+	0.030	4.35	0.60

TABLE C.17: Cosine-based evaluation - *Hybrid* SMbR results (cosine threshold = 0.85) - 2nd experimental setting

***Hierarchy* SMbR**	***Hybrid* SMbR**
0.059	0.060

TABLE C.18: Cosine-based evaluation - Average precision values for the Cb recommendation methods (cosine threshold = 0.85)- 2nd experimental setting

Appendix D

Evaluation of a Standalone Bayesian Recommender Algorithm

In this chapter we illustrate the evaluation of the Bayesian recommender algorithm introduced by [92] with the same experimental settings provided in Chapter 5. Specifically, in Tables D.1 - D.9 we present the average values of precision and recall, alongside with the (average) number of relevant POIs, Rec_r (the average number of relevant POIs among the 20 recommended) and *Recall UB* for all age groups, each of which comprises of 100 users who enter the system 30 times. Furthermore, we illustrate the average precision values for every $\langle m, n \rangle$ combination in Table D.10. We can observe that the outcomes are comparable with the respective of our Hybrid RS in Chapter 5.

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.879	0.064	0.073	272.5	17.5
26-35	0.906	0.065	0.072	275.3	18.1
36-45	0.895	0.065	0.073	274.0	17.9
46-55	0.867	0.064	0.074	269.9	17.3
56-67	0.871	0.063	0.073	272.9	17.4
67+	0.702	0.053	0.080	249.7	14.0

TABLE D.1: Bayesian RS performance; ($m = 4, n = 4$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.905	0.066	0.073	272.3	18.1
26-35	0.932	0.067	0.072	275.2	18.6
36-45	0.934	0.068	0.072	274.1	18.6
46-55	0.911	0.066	0.074	269.6	18.2
56-67	0.906	0.066	0.073	272.9	18.1
67+	0.736	0.056	0.080	248.9	14.7

TABLE D.2: Bayesian RS performance; ($m = 5, n = 5$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.897	0.065	0.073	272.2	17.9
26-35	0.927	0.067	0.072	275.6	18.5
36-45	0.928	0.067	0.072	274.3	18.5
46-55	0.897	0.065	0.074	269.7	17.9
56-67	0.902	0.065	0.073	273.0	18.0
67+	0.720	0.054	0.080	249.4	14.4

TABLE D.3: Bayesian RS performance; ($m = 6, n = 4$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.908	0.066	0.073	272.7	18.1
26-35	0.933	0.067	0.072	275.5	18.6
36-45	0.935	0.068	0.073	273.9	18.7
46-55	0.904	0.066	0.073	270.5	18.0
56-67	0.922	0.067	0.073	272.8	18.4
67+	0.731	0.055	0.080	249.5	14.6

TABLE D.4: Bayesian RS performance; ($m = 4, n = 6$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.892	0.065	0.073	273.0	17.8
26-35	0.923	0.067	0.072	275.5	18.6
36-45	0.929	0.067	0.073	273.9	18.5
46-55	0.896	0.065	0.074	269.9	17.9
56-67	0.894	0.065	0.073	273.0	17.8
67+	0.722	0.054	0.080	249.0	14.4

TABLE D.5: Bayesian RS performance; ($m = 5, n = 4$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.891	0.065	0.073	272.6	17.8
26-35	0.927	0.067	0.072	275.7	18.5
36-45	0.925	0.067	0.072	274.1	18.5
46-55	0.895	0.065	0.074	269.2	17.9
56-67	0.901	0.065	0.073	272.8	18.0
67+	0.723	0.055	0.080	249.4	14.4

TABLE D.6: Bayesian RS performance; ($m = 4, n = 5$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.916	0.066	0.073	272.2	18.3
26-35	0.948	0.068	0.072	275.2	18.9
36-45	0.942	0.068	0.072	274.1	18.8
46-55	0.918	0.067	0.074	270.0	18.3
56-67	0.915	0.062	0.073	272.7	18.3
67+	0.740	0.056	0.080	248.9	14.8

TABLE D.7: Bayesian RS performance; ($m = 6, n = 5$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.907	0.066	0.073	272.9	18.1
26-35	0.944	0.068	0.072	275.7	18.8
36-45	0.940	0.068	0.072	274.4	18.8
46-55	0.921	0.067	0.074	270.1	18.4
56-67	0.930	0.067	0.073	273.0	18.6
67+	0.759	0.057	0.080	249.0	15.1

TABLE D.8: Bayesian RS performance; ($m = 5, n = 6$)

Age Group	Precision	Recall	Recall UB	Relevant Items	Rec_r
17-25	0.908	0.066	0.073	272.3	18.1
26-35	0.944	0.068	0.072	275.2	18.8
36-45	0.945	0.069	0.073	273.7	18.9
46-55	0.926	0.068	0.073	270.3	18.5
56-67	0.925	0.067	0.073	272.8	18.5
67+	0.750	0.056	0.080	249.2	15.0

TABLE D.9: Bayesian RS performance; ($m = 6, n = 6$)

	$n = 4$	$n = 5$	$n = 6$
$m = 4$	0.853	0.877	0.888
$m = 5$	0.876	0.887	0.900
$m = 6$	0.878	0.896	0.899

TABLE D.10: Avg. precision per $\langle m, n \rangle$ pair (Bayesian RS)

Appendix E

Predicting Tweet Engagements

Within this chapter, we present a description of the methodology that we followed for the ACM RecSys Challenge 2021, which was arranged by Twitter. The objective of the competition was to anticipate four types of user interactions on Twitter, namely Like, Reply, Retweet, and Retweet with comment, predicated on preceding engagements on the platform. Our approach yields noteworthy insights from the dataset by means of generating a set of features. Ultimately, we employ gradient boosting trees to classify the various forms of user engagements. In conclusion, two metrics; relative cross entropy (RCE) and average-precision (AP), are utilized to assess the models for each type of user engagement.

E.1 Fairness

The Twitter 2021 challenge was characterized by a dual objective: firstly, to predict the likelihood of diverse forms of user engagements with a given set of tweets, leveraging a heterogeneous array of input data, and secondly, to ensure fair recommendations. Fairness, in this context, is a social construct that cannot be reduced to an optimization problem. In particular, recommendations should not be contingent on the popularity of Twitter users, who ought not to be disadvantaged for having a lesser degree of popularity on the platform.

E.2 Dataset Description

Information about the dataset is given by Table E.1 below:

Dataset	Training	Validation	Testing
Dataset size	309Gb	13.1Gb	6.9Gb
Records	667.7M	28.6M	14M
Unique tweets	261.1M	10.2M	9.9M
Unique engagers	24.1M	3.6M	3.5M
Unique engagees	34.3M	6.1M	6M

TABLE E.1: General information about the dataset

Table E.2 exhibits the classification of user engagements during the initial 24-hour period, along with the aggregate count (measured in millions) of each type of engagement within the training set. This suggests a preference among users for likes, followed by retweets and replies, and lastly, retweets with comments.

Engagement type	First 24 hours	First 3 weeks
Like	12.21M	264.2M
Retweet	2.7M	58.2M
Reply	0.83M	18.5M
RT with comment	0.2M	4.6M

TABLE E.2: Number of engagements in training set

E.3 Our Method's Results

Our method utilizes PySpark and specifically the GBT-Classifier module to tackle the prediction task. Specifically, we try to analyze the dataset and reduce its original size, encounter the fairness problem and take advantage of the gradient boosting trees to create the prediction models. Based on the final results of the competition leaderboad, our unsuccessful submission could have been ranked 16 or 17. Table E.3 presents the prediction outcomes of our method:

Engagement	Like	Reply	Retweet	Quote
AP	0.5486	0.0581	0.1756	0.0074
RCE	7.2393	-8.0678	3.4172	-86.3023

TABLE E.3: Results based on the unconstrained leaderboard

Bibliography

[1] Charu C Aggarwal et al. *Data mining: the textbook*. Vol. 1. Springer, 2015.

[2] Charu C Aggarwal et al. *Recommender systems*. Vol. 1. Springer, 2016.

[3] Sarabjot Singh Anand and Bamshad Mobasher. "Intelligent techniques for web personalization". In: *IJCAI Workshop on Intelligent Techniques for Web Personalization*. Springer. 2003, pp. 1–36.

[4] Konstantinos Babas, Georgios Chalkiadakis, and Evangelos Tripolitakis. "You Are What You Consume: A Bayesian Method for Personalized Recommendations". In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys '13. Hong Kong, China: Association for Computing Machinery, 2013, pp. 221–228. ISBN: 9781450324090. DOI: 10.1145/2507157.2507158. URL: https://doi.org/10.1145/2507157.2507158.

[5] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*. Vol. 463. ACM press New York, 1999.

[6] Nicola Barbieri, Gianni Costa, Giuseppe Manco, and Riccardo Ortale. "Modeling item selection and relevance for accurate recommendations: a bayesian approach". In: *2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*. Ed. by Bamshad Mobasher, Robin D. Burke, Dietmar Jannach, and Gediminas Adomavicius. ACM, 2011, pp. 21–28. DOI: 10.1145/2043932.2043941. URL: https://doi.org/10.1145/2043932.2043941.

[7] Robert Bell, Yehuda Koren, and Chris Volinsky. "Modeling relationships at multiple scales to improve accuracy of large recommender systems". In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007, pp. 95–104.

[8] Robert M Bell and Yehuda Koren. "Scalable collaborative filtering with jointly derived neighborhood interpolation weights". In: *Seventh IEEE international conference on data mining (ICDM 2007)*. IEEE. 2007, pp. 43–52.

[9] James Bennett, Stan Lanning, et al. "The netflix prize". In: *Proceedings of KDD cup and workshop*. Vol. 2007. New York. 2007, p. 35.

[10] Idir Benouaret and Dominique Lenne. "Personalizing the museum experience through context-aware recommendations". In: *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE. 2015, pp. 743–748.

[11] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. "Mediation of user models for enhanced personalization in recommender systems". In: *User Modeling and User-Adapted Interaction* 18.3 (2008), pp. 245–286.

[12] Daniel Billsus and Michael Pazzani. "Learning probabilistic user models". In: *UM97 Workshop on Machine Learning for User Modeling*. Citeseer. 1997.

[13] Daniel Billsus and Michael J Pazzani. "A hybrid user model for news story classification". In: *Um99 user modeling*. Springer, 1999, pp. 99–108.

[14] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[15] Basilis Boutsinas and Thomas Papastergiou. "On clustering tree structured data with categorical nature". In: *Pattern Recognition* 41.12 (2008), pp. 3613–3623.

[16] Robin Burke. "Hybrid recommender systems: Survey and experiments". In: *User modeling and user-adapted interaction* 12.4 (2002), pp. 331–370.

[17] Robin Burke. "Hybrid web recommender systems". In: *The adaptive web* (2007), pp. 377–408.

[18] John Canny. "Collaborative filtering with privacy via factor analysis". In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. 2002, pp. 238–245.

[19] M Emre Celebi, Hassan A Kingravi, and Patricio A Vela. "A comparative study of efficient initialization methods for the k-means clustering algorithm". In: *Expert systems with applications* 40.1 (2013), pp. 200–210.

[20] Georgios Chalkiadakis, Ioannis Ziogas, Michail Koutsmanis, Errikos Streviniotis, Costas Panagiotakis, and Harris Papadakis. "A novel hybrid recommender system for the tourism domain". In: *Algorithms* 16.4 (2023), p. 215.

[21] Kinjal Chaudhari and Ankit Thakkar. "A comprehensive survey on travel recommender systems". In: *Archives of Computational Methods in Engineering* 27.5 (2020), pp. 1545–1571.

[22] Yan-Ying Chen, An-Jung Cheng, and Winston H Hsu. "Travel recommendation by mining people attributes and travel group types from community-contributed photos". In: *IEEE Transactions on Multimedia* 15.6 (2013), pp. 1283–1295.

[23] Brett Chudoba. "How much time are respondents willing to spend on your survey". In: *Retrieved October 11, 2015, from www. surveymonkey. com/blog/en/blog/2011/02* 14 (2011).

[24] David R Cox. "The regression analysis of binary sequences". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 21.1 (1959), pp. 238–238.

[25] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. "Indexing by latent semantic analysis". In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407.

[26] Christiane Fellbaum. "WordNet". In: *Theory and applications of ontology: computer applications*. Springer, 2010, pp. 231–243.

[27] Junmei Feng, Zhaoqiang Xia, Xiaoyi Feng, and Jinye Peng. "RBPR: A hybrid model for the new user cold start problem in recommender systems". In: *Knowledge-Based Systems* 214 (2021), p. 106732.

[28] Gerhard Fischer. "User modeling in human–computer interaction". In: *User modeling and user-adapted interaction* 11.1 (2001), pp. 65–86.

[29] Dominic Girardi, Sandra Wartner, Gerhard Halmerbauer, Margit Ehrenmüller, Hilda Kosorus, and Stephan Dreiseitl. "Using concept hierarchies to improve calculation of patient similarity". In: *Journal of biomedical informatics* 63 (2016), pp. 66–73.

[30] Charles R Goeldner and JR Brent Ritchie. *Tourism principles, practices, philosophies*. John Wiley & Sons, 2007.

[31] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. "Using collaborative filtering to weave an information tapestry". In: *Communications of the ACM* 35.12 (1992), pp. 61–70.

[32] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. "Eigentaste: A constant time collaborative filtering algorithm". In: *Information Retrieval* 4.2 (2001), pp. 133–151.

[33] Djamel Guessoum, Moeiz Miraoui, and Chakib Tadj. "A modification of wu and palmer semantic similarity measure". In: *The Tenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*. 2016, pp. 42–46.

[34] Clare A Gunn. "Tourism Planning, Crane, Russak & Company". In: *Inc., New York* (1979).

[35] Conor Hayes and Pádraig Cunningham. "Smart radio—community based music radio". In: *Knowledge-Based Systems* 14.3-4 (2001), pp. 197–201.

[36] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. "Evaluating collaborative filtering recommender systems". In: *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), pp. 5–53.

[37] Robert C Holte and John Ng Yuen Yan. "Inferring what a user is not interested in". In: *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer. 1996, pp. 159–171.

[38] Cho-Jui Hsieh, Nagarajan Natarajan, and Inderjit Dhillon. "PU learning for matrix completion". In: *International conference on machine learning*. Proceedings of Machine Learning Research. 2015, pp. 2445–2453.

[39] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets". In: *2008 Eighth IEEE international conference on data mining*. Ieee. 2008, pp. 263–272.

[40] Paul Jaccard. "The distribution of the flora in the alpine zone. 1". In: *New phytologist* 11.2 (1912), pp. 37–50.

[41] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[42] Mohamed Elyes Ben Haj Kbaier, Hela Masri, and Saoussen Krichen. "A personalized hybrid tourism recommender system". In: *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*. Ieee. 2017, pp. 244–250.

[43] 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*. 2008, pp. 426–434.

[44] Yehuda Koren. "The bellkor solution to the netflix grand prize". In: *Netflix prize documentation* 81.2009 (2009), pp. 1–10.

[45] Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems". In: *Computer* 42.8 (2009), pp. 30–37.

[46] G Kurtenbach, Abigail J Sellen, and William AS Buxton. "Some articulatory and cognitive aspects of marking menus: an empirical study". In: *Human Computer Interaction* 8.2 (1993), pp. 1–23.

[47] Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, and Noor Akhmad Setiawan. "Cosine similarity to determine similarity measure: Study case in online essay assessment". In: *2016 4th International Conference on Cyber and IT Service Management*. 2016, pp. 1–6. DOI: 10.1109/CITSM.2016.7577578.

[48] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. "Personalized tour recommendation based on user interests and points of interest visit durations". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.

[49] Greg Linden, Brent Smith, and Jeremy York. "Amazon. com recommendations: Item-to-item collaborative filtering". In: *IEEE Internet computing* 7.1 (2003), pp. 76–80.

[50] Steve Lohr. "A $1 million research bargain for Netflix, and maybe a model for others". In: *The New York Times* 22 (2009).

[51] Phillip W. Lord, Robert D. Stevens, Andy Brass, and Carole A. Goble. "Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation". In: *Bioinformatics* 19.10 (2003), pp. 1275–1283.

[52] Tariq Mahmood and Francesco Ricci. "Improving recommender systems with adaptive conversational strategies". In: *Proceedings of the 20th ACM conference on Hypertext and hypermedia*. 2009, pp. 73–82.

[53] Andrew McCallum, Kamal Nigam, et al. "A comparison of event models for naive bayes text classification". In: *AAAI-98 workshop on learning for text categorization*. Vol. 752. 1. Madison, WI. 1998, pp. 41–48.

[54] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra. "Explore, exploit, and explain: personalizing explainable recommendations with bandits". In: *Proceedings of the 12th ACM conference on recommender systems*. 2018, pp. 31–39.

[55] Frank McSherry and Ilya Mironov. "Differentially private recommender systems: Building privacy into the netflix prize contenders". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 627–636.

[56] Ram Krishn Mishra, Siddhaling Urolagin, and J Angel Arul Jothi. "Sentiment analysis for poi recommender systems". In: *2020 seventh international conference on information technology trends (ITT)*. IEEE. 2020, pp. 174–179.

[57] Tom M Mitchell and Tom M Mitchell. *Machine learning*. Vol. 1. 9. McGraw-hill New York, 1997.

[58] Dunja Mladenic. "Text-learning and related intelligent agents: a survey". In: *IEEE intelligent systems and their applications* 14.4 (1999), pp. 44–54.

[59] Miquel Montaner, Beatriz López, and Josep Lluís De La Rosa. "A taxonomy of recommender agents on the internet". In: *Artificial intelligence review* 19.4 (2003), pp. 285–330.

[60] Costas Panagiotakis, Evangelia Daskalaki, Harris Papadakis, and Paraskevi Fragopoulou. "The tourist trip design problem with POI categories via an Expectation-Maximization based method". In: *RecSys Workshop on Recommenders in Tourism (RecTour 2022)*. 2022.

[61] Harris Papadakis, Costas Panagiotakis, Paraskevi Fragopoulou, Georgios Chalkiadakis, Errikos Streviniotis, Ioannis-Panagiotis Ziogas, Michail Koutsmanis, and Panagiotis Bariamis. "Visit Planner: A Personalized Mobile Trip Design Application based on a Hybrid Recommendation Model". In: (2023).

[62] Harris Papadakis, Antonis Papagrigoriou, Costas Panagiotakis, Eleftherios Kosmas, and Paraskevi Fragopoulou. "Collaborative filtering recommender systems taxonomy". In: *Knowledge and Information Systems* 64.1 (2022), pp. 35–74.

[63] Michael J Pazzani and Daniel Billsus. "Content-based recommendation systems". In: *The adaptive web*. Springer, 2007, pp. 325–341.

[64] Michael J Pazzani, Jack Muramatsu, Daniel Billsus, et al. "Syskill & Webert: Identifying interesting web sites". In: *AAAI/IAAI, Vol. 1*. 1996, pp. 54–61.

[65] Catia Pesquita, Daniel Faria, Andre O Falcao, Phillip Lord, and Francisco M Couto. "Semantic similarity in biomedical ontologies". In: *PLoS computational biology* 5.7 (2009), pp. 1–12.

[66] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. "Development and application of a metric on semantic nets". In: *IEEE transactions on systems, man, and cybernetics* 19.1 (1989), pp. 17–30.

[67] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. "BPR: Bayesian personalized ranking from implicit feedback". In: *arXiv preprint arXiv:1205.2618* (2012).

[68] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. "Grouplens: An open architecture for collaborative filtering of netnews". In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 1994, pp. 175–186.

[69] Paul Resnick and Hal R Varian. "Recommender systems". In: *Communications of the ACM* 40.3 (1997), pp. 56–58.

[70] Philip Resnik. "Using information content to evaluate semantic similarity in a taxonomy". In: *arXiv preprint cmp-lg/9511007* (1995).

[71] Francesco Ricci, Dario Cavada, Nader Mirzadeh, Adriano Venturini, et al. "Case-based travel recommendations". In: *Destination recommendation systems: behavioural foundations and applications* (2006), pp. 67–93.

[72] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook". In: *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[73] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Recommender systems: introduction and challenges". In: *Recommender systems handbook*. Springer, 2015, pp. 1–34.

[74] Elaine Rich. "User modeling via stereotypes". In: *Cognitive science* 3.4 (1979), pp. 329–354.

[75] Masoumeh Riyahi and Mohammad Karim Sohrabi. "Providing effective recommendations in discussion groups using a new hybrid recommender system based on implicit ratings and semantic similarity". In: *Electronic Commerce Research and Applications* 40 (2020), p. 100938.

[76] Joseph Rocchio. "Relevance feedback in information retrieval". In: *The Smart retrieval system-experiments in automatic document processing* (1971), pp. 313–323.

[77] David James Rogers and Tyrus T. Tanimoto. "A Computer Program for Classifying Plants." In: *Science* 132 3434 (1960), pp. 1115–8.

[78] Deepjyoti Roy and Mala Dutta. "A systematic review and research perspective on recommender systems". In: *Journal of Big Data* 9.1 (2022), p. 59.

[79] M. Ružička. "Anwendung mathematisch-statistischer Methoden in der Geobotanik (Synthetische Bearbeitung von Aufnahmen)". In: *Biológia, Bratislava* 13: (1958), pp. 647–661.

[80] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering". In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 791–798.

[81] Gerard Salton. "Automatic text processing: The transformation, analysis, and retrieval of". In: *Reading: Addison-Wesley* 169 (1989).

[82] Gerard Salton and Michael J McGill. *Introduction to modern information retrieval*. Mcgraw-Hill, 1983.

[83] Manash Sarkar, Arup Roy, Maroi Agrebi, and Hameed AlQaheri. "Exploring New Vista of Intelligent Recommendation Framework for Tourism Industries: An Itinerary through Big Data Paradigm". In: *Information* 13.2 (2022), p. 70.

[84] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. "Collaborative filtering recommender systems". In: *The adaptive web*. Springer, 2007, pp. 291–324.

[85] Fabrizio Sebastiani. "Machine learning in automated text categorization". In: *ACM computing surveys (CSUR)* 34.1 (2002), pp. 1–47.

[86] Mete Sertkan, Julia Neidhardt, and Hannes Werthner. "PicTouRe - A Picture-Based Tourism Recommender". In: *Fourteenth ACM Conference on Recommender Systems*. RecSys '20. Virtual Event, Brazil: Association for Computing Machinery, 2020, pp. 597–599. ISBN: 9781450375832. DOI: 10.1145/3383313.3411526. URL: https://doi.org/10.1145/3383313.3411526.

[87] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. "An MDP-based recommender system." In: *Journal of Machine Learning Research* 6.9 (2005).

[88] Upendra Shardanand and Pattie Maes. "Social information filtering: Algorithms for automating "word of mouth"". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1995, pp. 210–217.

[89] Manjula K Shenoy, KC Shet, and U Dinesh Acharya. "A new similarity measure for taxonomy based on edge counting". In: *International Journal of Web & Semantic Technology* 3.4 (2012), p. 23.

[90] Thabet Slimani. "Description and evaluation of semantic similarity measures approaches". In: *arXiv preprint arXiv:1310.8059* (2013).

[91] Errikos Streviniotis and Georgios Chalkiadakis. "Multiwinner election mechanisms for diverse personalized Bayesian recommendations for the tourism domain". In: *2022 Workshop on Recommenders in Tourism, RecTour*. 2022.

[92] Errikos Streviniotis and Georgios Chalkiadakis. "Preference Aggregation Mechanisms for a Tourism-Oriented Bayesian Recommender". In: *PRIMA 2022: Principles and Practice of Multi-Agent Systems: 24th International Conference, Valencia, Spain, November 16–18, 2022, Proceedings*. Springer. 2022, pp. 331–346.

[93] Michael Strube and Simone Paolo Ponzetto. "WikiRelate! Computing semantic relatedness using Wikipedia". In: *Association for the Advancement of Artificial Intelligence*. Vol. 6. 2006, pp. 1419–1424.

[94] Jianing Sun, Wei Guo, Dengcheng Zhang, Yingxue Zhang, Florence Regol, Yaochen Hu, Huifeng Guo, Ruiming Tang, Han Yuan, Xiuqiang He, et al. "A framework for recommending accurate and diverse items using bayesian graph convolutional neural networks". In: *26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 2030–2039.

[95] Evangelos Tripolitakis and Georgios Chalkiadakis. "Probabilistic Topic Modeling, Reinforcement Learning, and Crowdsourcing for Personalized Recommendations". In: *Multi-Agent Systems and Agreement Technologies - 14th European Conference, EUMAS 2016, and 4th International Conference, AT 2016, Valencia, Spain, December 15-16, 2016, Revised Selected Papers*. Ed. by Natalia Criado Pacheco, Carlos Carrascosa, Nardine Osman, and Vicente Julián Inglada. Vol. 10207. Lecture Notes in Computer Science. Springer, 2016, pp. 157–171. DOI: 10.1007/978-3-319-59294-7_14. URL: https://doi.org/10.1007/978-3-319-59294-7%5C_14.

[96] Artem Umanets, Artur Ferreira, and Nuno Leite. "GuideMe–A tourist guide with a recommender system and social interaction". In: *Procedia Technology* 17 (2014), pp. 407–414.

[97] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. "Graph neural networks in recommender systems: a survey". In: *ACM Computing Surveys* 55.5 (2022), pp. 1–37.

[98] Zhibiao Wu and Martha Palmer. "Verb semantics and lexical selection". In: *arXiv preprint cmp-lg/9406033* (1994).

[99] Jun Zeng, Feng Li, Haiyang Liu, Junhao Wen, and Sachio Hirokawa. "A restaurant recommender system based on user preference and location in mobile environment". In: *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*. IEEE. 2016, pp. 55–60.

[100] Xiaoyao Zheng, Yonglong Luo, Zhiyun Xu, Qingying Yu, and Lin Lu. "Tourism destination recommender system for the cold start problem". In: *KSII Transactions on Internet and Information Systems (TIIS)* 10.7 (2016), pp. 3192–3212.

[101] Ioannis Panagiotis Ziogas, Errikos Streviniotis, Harris Papadakis, and Georgios Chalkiadakis. "Content-based recommendations using similarity distance measures with application in the tourism domain". In: *Proceedings of the 12th Hellenic Conference on Artificial Intelligence*. 2022, pp. 1–10.