Beaccomender : A Sensor-based Recommender System for Smart Phones

Stergios Chairistanidis (517)

Advisor : Dr. Panagiotis Symeonidis
Prof Dr. Y. Manolopoulos

First Referee: Prof. Dr. Ioannis Stamelos
Second Referee: Prof. Dr. Eugenia Petridou
Abstract

In the past few years, there is a rapid evolution of different kinds of networks over the Internet (i.e. users-products network of an eshop, interrelated web pages through hyperlinks, etc.), even though, we are mainly aware of social networks, such as Facebook and Twitter. The problem of measuring “similarity” of objects arises in all these types of networks. We decided to cope with the “Link Prediction” problem. We devised a method called “Omni-Rank”, which is an extension of the well-known Simrank algorithm, and combined it with the Collaborative Filtering technique in the recommender systems’ domain. Furthermore, in this thesis we aim to bring closer the digital stores(eshops) with the conventional ones by exploiting iBeacon technology and smartphones, which, along with the help of mobile apps, would take advantage of geographical position of users and be benefited from their physical context (time, temperature, motion, etc.) to leverage the accuracy of recommendations.

Our main aim in this thesis is to construct such a system, which would combine the preferences of users resulting from their behaviour on a website with their preferences resulting from a visit to a local store or a local shopping center. The combination of the above preferences would result in enhanced personalised recommendations which would be more accurate and include a variety of items. Concerning our proposed algorithm, our goal was to show that it could provide better results in terms of link prediction and similar object search. We have conducted an experimental comparison of our method against two other state-of-the-art link prediction algorithms (Simrank and P-Rank) on two real-world datasets (Geosocialrec and Hetrec Movielens 2011). Our experimental results have shown that our method is superior than its competitors in terms of recommendations’ accuracy. We finally propose some potential business exploitations of our proposed recommender system.

Keywords: Link prediction, Data mining, Recommender Systems, Simrank, Bipartite Networks, implicit ratings, Web 2.0, link similarity, iBeacon, context-aware recommendation.
Περίληψη

Τα προηγούμενα χρόνια, υπάρχει μια παγκόσμια ανάπτυξη διαφορετικών ειδών δικτύων στο Διαδίκτυο (π.χ. χρήστες - προϊόντα σε ένα ηλεκτρονικό κατάστημα, συσχετιζόμενες ιστοσελίδες μέσω υπερσυνδέσμων, κτλ.), πέρα από τα κοινωνικά δικτύα που είναι αυτά που γνωρίζουμε κυρίως, όπως το Facebook και το Twitter. Σε όλους αυτούς τους τύπους δικτύων εγείρεται το πρόβλημα μέτρησης της "ομοιότητας" των αντικειμένων. Εμείς αποφασίσαμε να αντιμετωπίσουμε αυτό το πρόβλημα με το "Link Prediction". Κατασκευάσαμε μια μέθοδο, την OmniRank, η οποία είναι μια επέκταση του πολύ γνωστού αλγόριθμου SimRank, και τη συνδυάσαμε με την τεχνική του Συνεργατικού Φίλτραρισμάτος. Επιπλέον, στη συγκεκριμένη διπλωματική εργασία, σκοπεύουμε να φέρουμε πιο κοντά τα ψηφιακά καταστήματα (eshops) με τα συμβατικά χρησιμοποιώντας την τεχνολογία "iBeacon" και τα έξυπνα τηλέφωνα, τα οποία, μαζί με τη βοήθεια των εφαρμογών κινητών συσκευών, θα αξιοποιήσουν τη γεωγραφική θέση των χρηστών και θα επωφεληθούν από τα στοιχεία του περιβάλλοντός τους (χρόνος, θερμοκρασία, κίνηση) για να βελτιώσουν την ακρίβεια των συστάσεων.

Ο κύριος στόχος μας σε αυτή την διπλωματική εργασία είναι να κατασκευάσουμε ένα τέτοιο σύστημα, που θα συνδυάζει τις προτιμήσεις των χρηστών οι οποίες θα προκύπτουν από τη συμπεριφορά τους όταν επισκέπτονται μία ιστοσελίδα και τις προτιμήσεις τους από μία επίσκεψη σε ένα τοπικό κατάστημα ή εμπορικό κέντρο. Ο συνδυασμός των παραπάνω προτιμήσεων θα έχει σαν αποτέλεσμα βελτιωμένες και πιο προσωποποιημένες συστάσεις οι οποίες θα είναι πιο ακριβείς και θα περιλαμβάνουν πληθώρα προϊόντων.

Αναφορικά με τον προτεινόμενο αλγόριθμο, ο σκοπός μας είναι να δείξουμε ότι μπορεί να παρέχει καλύτερα αποτελέσματα σχετικά με την αναζήτηση όμοιων αντικειμένων. Στην παρούσα διπλωματική εργασία έχουμε διενεργήσει μία πειραματική σύγκριση της μέθοδος μας με δύο άλλους πολύ γνωστούς αλγόριθμους (SimRank και P-Rank) πάνω σε δύο πραγματικά σύνολα δεδομένων (Geosocialrec και Hetrec Movielens 2011). Τα πειραματικά αποτελέσματα μας δείχνουν ότι η μέθοδος μας είναι ανώτερη από τους ανταγωνιστές της σε σχέση με την ακρίβεια συστάσεων. Τέλος, προτείνουμε κάποιες πιθανές εμπορικές αξιοποιήσεις του προτεινόμενου συστήματος συστάσεων.

Λέξεις κλειδιά: Link prediction, Data mining, Recommender Systems, Simrank, Bipartite Networks, implicit ratings, Web 2.0, link similarity, iBeacon, context-aware recommendation.
# Contents

List of Figures vi

List of Tables viii

1 Introduction 2
   1.1 Problem Definition 2
   1.2 Outline 3

2 Beacons and Indoor location-based services 6
   2.1 LBS and indoor positioning 6
   2.2 Bluetooth Low Energy 6
   2.3 Beacons 8
   2.4 Estimote 9
   2.5 Beacons and location-based services 11

3 Recommender Systems and Link Similarity algorithms 14
   3.1 Definition 14
   3.2 Recommendation Process 15
   3.3 Recommender System Classifications 16
      3.3.1 Content-based Filtering 17
      3.3.2 Collaborative Filtering 17
      3.3.3 Hybrid Recommender Systems 18
   3.4 Recommender System Examples 19
   3.5 Sensor-based Recommendations 20

4 Link Similarity algorithms 24
   4.1 Introduction to graphs 24
   4.2 SimRank 25
   4.3 P-Rank 27
   4.4 OmniRank 30

5 Beacommender System 34
   5.1 Website 35
      5.1.1 Homepage 35
      5.1.2 Product’s Page 36
   5.2 Web Database 38
      5.2.1 User-based CF recommendation example 40
   5.3 Web Server 40
   5.4 Mobile iOS Application 42
      5.4.1 Storing beacon preferences 43
      5.4.2 Get Discounts 45
5.4.3 Beacon Recommendations ........................................ 46

6 Experimental Evaluation and Results ......................... 48
   6.1 GeoSocialRec dataset ............................................. 48
   6.2 HetRec MovieLens dataset ...................................... 48
   6.3 Experimental Setup .............................................. 49
      6.3.1 Evaluation Method ......................................... 49
      6.3.2 Performance Measures ..................................... 50
   6.4 Sensitivity Analysis of the OmniRank Algorithm ............ 52
   6.5 Comparison with other methods .............................. 54

7 Conclusion and exploitation of Beacommender ............... 58

A Web Database ...................................................... 64

B Web Server ......................................................... 66

C Evaluation Process Code ........................................ 68
## List of Figures

2.1 BLE topology[10] .................................................. 8  
2.2 Beacon's main parts ............................................ 9  
2.3 Estimote - iBeacon logo ...................................... 10  
2.4 Nivea's campaign with iBeacons ............................ 11  
2.5 iStorm app utilizes iBeacon for increased customer experience .... 12  
2.6 Virgin Atlantic iBeacon implementation at the Heathrow airport .... 12  
3.1 Recommendation process - adopted from [20] ............... 16  
3.2 Amazon recommendations ..................................... 19  
3.3 Netflix recommendations ...................................... 19  
3.4 LinkedIn recommendations .................................... 20  
3.5 Context-aware recommendations based on beacons - Example .... 22  
4.1 A visual representation of our toy example(User-Movie-Tags) ........ 25  
5.1 Beacommender System architecture ......................... 34  
5.2 Beacommender.eu Homepage ................................ 35  
5.3 Products' ratings ............................................. 36  
5.4 Related Products ............................................. 37  
5.5 Recommended Products ....................................... 37  
5.6 Recommendation Results page ............................... 38  
5.7 Tables of database ........................................... 39  
5.8 Product ID and product description .......................... 41  
5.9 A json object example ...................................... 42  
5.10 iOS Application Tabs - Homepage and registration .......... 42  
5.11 iOS Application Tabs - Store sections and products in them .... 43  
5.12 Xcode environment .......................................... 44  
5.13 DidExitRegion Function .................................... 45  
5.14 iOS Application Tabs - Store sections and products in them .... 46  
5.15 iOS Application Tabs - Product Details and Recommendations .... 47  
6.1 Parameters of evaluation in java ............................ 49  
6.2 Typical Precision-Recall curve ................................ 51  
6.3 Analysis of Omnirank performance metrics based on different percentages of training set for both datasets. We see the different precision scores (a) and (b), the different recall scores (c) and (d) and the F1-Score (e), (f) for Geosocialrec and Hetrec datasets .................................................. 53  
6.4 Analysis of Omnirank performance metrics based on different percentages of observed edges between (a) locations for Geosocialrec dataset and (b) movie genres for Hetrec dataset. ............................... 54
6.5 Comparing Simrank, P-Rank and OmniRank performance in term of Precision and Recall at top-N recommended items (a) on GeoSocial dataset and (b) on HetRec dataset. .................................................. 55
6.6 Algorithms’ comparison on F1-metric vs. different values of top-N Recommended Locations - GeosocialRec dataset ........................................ 56
6.7 Algorithms’ comparison on F1-metric vs. different values of Recommended movies - HetRec dataset .................................................. 56
A.1 Ratings stored in the database ............................................. 64
A.2 Beacon ratings stored in the database ................................. 65
# List of Tables

2.1 Indoor technologies for LBS ........................................... 7  
3.1 Term Definition ......................................................... 21  
3.2 Mapping preference between user and item .......................... 21  
4.1 Structural Similarity Scores for SimRank algorithm ............... 27  
4.2 Structural Similarity Scores ........................................... 28  
4.3 Structural Similarity Scores ........................................... 31  
6.1 Results of OmniRank on Geocosial and Hetrec Movielens datasets based on the percentage of the training set ......................... 52  
6.2 Results of OmniRank on Geocosial and Hetrec Movielens datasets based on the percentage of the fraction of edges observed ............. 54  
6.3 Algorithms’ comparison on GeoSocialRec data set with different number of Top-N recommendations ................................. 55  
6.4 Algorithms’ comparison on HetRec data sets with different number of Top-N recommendations ........................................... 56
Chapter 1

Introduction

1.1 Problem Definition

In Web, the large volumes of data produced by users and their online interactions with items, can be exploited by the emerging research area of data mining, called Web Mining. One basic technique regarding web mining is web structure mining. Specifically, it is the process of using graph theory to analyze the node and connection structure of a web site. Web sites use graphs in their network web structure (i.e., users, products, time). That is, web sites and e-Shops use graphs, where different types of vertices depict users/items, and different types of edges that connect these vertices represent their relationships. Nowadays, it has been identified that users suffer from the overload of information in Web 2.0. Thus, there is a great potential for web mining tools to provide solutions that filter this information.

Nowadays, there is a huge amount of information available to everyone. When someone is about to make a selection and he wants to maximize the utilization of it, he may retrieve and use all this information. The selection refers to either buying a product against another one or listening to a specific song or watching a particular movie or eating an unknown food, etc. As you can understand, in almost every decision nowadays you have to select something and reject something else. When a person is about to make a selection, he wants to make the best choice to maximize his satisfaction.

But the question rises: How can he retrieve the information he wants and how can he find out if this info is relevant to him? Imagine, for example, someone who wants to taste a food at a restaurant and receives various ratings about a specific plate. These ratings come from various persons who s/he may not know. In the real world, you receive recommendations based on the “word of mouth”. We usually know the persons who recommend something to us. We may also know their tastes and if these tastes match ours. In the Web, it is infeasible to know everyone who rates a product or an item. So, a recommender system, which would propose to the target user appropriate items, needs to be developed. Furthermore, the complexity of today’s networks is increased in comparison with the previous years. In order to cope with this problem, an algorithm which would easily compute the similar persons needs to be developed.

Based on the problems we discussed earlier, we decided to create a recommender system, which would help people decide the items to buy in a specific store. As Ricci et al. [1] denoted, recommender systems are a subclass of information filtering systems that seek to predict the ‘rating’ or ‘preference’ that a user would give to an item. This rec-
ommender system would take into account the user’s previous ratings and the product currently viewing. One of the first online recommender systems and maybe still the most productive and profitable is Amazon’s [2]. Moreover, we thought that it would be interesting if we could combine the user’s ratings over the Internet along with his behaviour inside a conventional store. The longer someone remains in a shop, the more possible it is for someone to look for something inside that store and thus resulting in a positive rating for that store. What is more, in comparison with another store, there may be different ratings based on the time someone remains inside each store. The same logic may apply inside a store among the various store sections. We decided to accomplish the previous tasks using iBeacon technology, recently published by Apple company. This technology includes some sensors, called beacons, which can detect the customers’ presence and with the help of mobile applications, they can be used to store the time someone remains on that region.

We also wanted to discover new ways to capture similarity among users and items to leverage the precision that current link similarity measures achieve. The “link prediction problem”, as denoted by Liben-Nowell and Kleinberg [3], exists in social networks, as well as, every other network we may form (protein-protein networks, web link structure, etc.). With the ubiquity of information networks and their broad applications, the issue of similarity computation between entities of an information network arises and attracts a lot of research attention [4].

There is a lot of research work in the domain of link prediction in graphs. We can distinguish two different algorithms for calculating the similarity between nodes in such a network. The first one is SimRank [5] and the second one is P-Rank [4]. Our newly proposed algorithm is called ‘OmniRank’, which extends the properties/capabilities of the previous two algorithms. In this thesis, we also develop a recommender system, which includes a website, a database, a web server and a mobile application for iOS devices. This system is denoted as “Beacommender System”, because it combines beacon technology with recommendations. That is, it provides context-aware recommendations based on the sensor technology and smart phone devices.

1.2 Outline

This postgraduate thesis is structured in 6 chapters. In the following chapters we will describe in more details each aspect of our ‘Beacommender System’ and our link similarity method 'OmniRank'. In Chapter 1, we define location based services (LBS). LBS are very popular today because of the wide spread and usage of mobile devices, especially smart phones, and the utilisation of GPS technology. Due to the fact that people tend to spend more time indoors, we need a technology which will help smart devices to locate position when GPS is not available. We describe various such technologies and pick iBeacon, a Bluetooth based indoor LBS. We also explain how this technology applies and one of the main beacon producers, Estimote. Estimote also has published and SDK for iOS devices which we utilised when developing ‘Beacommender System’. In Chapter 2, we describe a recommender system (RS). We talk about its three main classifications:

- Content-based Filtering
Chapter 1: Introduction

- Collaborative Filtering
- Hybrid Recommender Systems

We also present some of the most well-known RSs available on the Internet and we define an example where we describe how we have built our System. In Chapter 3, we make an introduction on graphs and then we mention the two most widely used graph methods for measuring node similarity, SimRank and P-Rank. Next, we define our method, OmniRank, which is an extension of the former two. We also explain how these algorithms work. In Chapter 4, we present 'Be recommender System'. We explain everything in detail and we provide various images of all our main elements in the system’s architecture (web site, database, servlet, app, etc.). In chapter 5, we discuss experimetal results of our proposed method, OmniRank. Firstly, please notice that the datasets we are going to use are two real life datasets, i.e., “Geosocialrec” and “HetRec Movielens 2011”. We also state the evaluation method and the performance metrics we are going to use. The whole evaluation project was fully built in Java - Eclipse. We split randomly the training and the test set, so we decided to run each evaluation five times to minimize the randomisation effect, which could lead to biased results. In chapter 6, we discuss the possible commercial exploitation of our “Be recommender System”. We are also stating a scenario where our system would be very useful and we are making suggestions how we could improve even more our system.

1 http://delab.csd.auth.gr/~symeon/
2 http://grouplens.org/datasets/hetrec-2011/
Chapter 2

Beacons and Indoor location-based services

2.1 LBS and indoor positioning

Advances in Internet, network technology and the rapidly growing number of mobile personal devices have resulted in the fast growth of Location-based Services (LBS). A location-based service can be described as an application that is dependent on a certain location [6]. LBS provide the mobile clients personalized services according to their current location. They also open a new area for developers, cellular service network operators, and service providers to develop and provide value-added services: advising clients of current traffic conditions, providing routing information, helping the users to find nearby shopping malls [7]. The core of any location based service is the positioning technology involved in tracking the current location of the user. Positioning technology refers to the various approaches used to approximate the location of a mobile device and thereby also its user. Nowadays, almost everyone uses smart devices and moreover smartphones. In order to find their exact location, smartphones use the GPS technology mainly. GPS stands for global positioning system. The main drawback of GPS technology is that it cannot show exact location when applied inside buildings. Due to the fact that people tend to spend even more time indoors when leaving homes (Shopping centres, large stores, supermarkets, restaurants), other technologies were applied to determine people location and provide them with appropriate services. The three commonest are Wi-Fi, RFID and Bluetooth, as shown in Table 2.1. WiFi measures the signal of all devices containing wireless adapters and there are estimates as to where the device is, based on network topology. Radio Frequency IDentification (RFID) technology uses radio frequency to achieve the goal of recognition and positioning by non-contact two-way data communication. The Bluetooth technology locates the objects by measuring the received signal strength, thus determining the distance between the object propagating a signal and the object that receives it [8].

2.2 Bluetooth Low Energy

BLE is designed for transmission of small amount of data and ultra-low power consumption. As Table 2.1 shows, BLE/iBeacon technology is not only less power consuming than
the others, but it also achieves higher accuracy and the devices using this technology last longer. The reason is that it uses the idea of “Get what you want when you want” and this allows the devices to be awoken only when are asked for data. In this way, devices consume less power.

<table>
<thead>
<tr>
<th></th>
<th>WiFi</th>
<th>RFID</th>
<th>iBeacon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage</td>
<td>50m</td>
<td>10m</td>
<td>50m</td>
</tr>
<tr>
<td>Cost</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>High</td>
<td>Low</td>
<td>Ultra Low</td>
</tr>
<tr>
<td>Battery Life</td>
<td>Several Days</td>
<td>1 - 2 years</td>
<td>2 - 3 years</td>
</tr>
<tr>
<td>Position Accuracy</td>
<td>2m - 3m</td>
<td>1m - 2m</td>
<td>&lt;1m</td>
</tr>
</tbody>
</table>

Table 2.1 Indoor technologies for LBS

In 2013, Apple introduced a protocol developed by the same company at Apple Worldwid Developers Conference, called iBeacon. This protocol can be exploited by hardware transmitters (typically called beacons) which broadcast a unique identifier to the space and nearby portable devices can receive this signal and perform an action. It basically acts as a trigger. These devices conform to BLE (Bluetooth Low Energy) protocol, originally introduced by Nokia, which is part of Bluetooth 4.0 [9]. BLE is quite efficient in terms of power consumption, rendering smart devices less battery-consuming and users more willing to enable bluetooth on their devices. The first commercial device which supported BLE protocol was iPhone 4s, but since then, every new device containing someone of the most popular operating systems (Android, iOS, Windows) supports the protocol. Bluetooth Low Energy devices can be in different operating states and roles depending on its function. Therefore, the possible states are the following:

- **Standby**: Does not transmit or receive packets
- **Advertising**: Broadcasts advertisements in advertising channels
- **Scanning**: Looks for advertisers
- **Initiating**: Initiates connection to advertiser
- **Connection**:
  - **Master Role**: Communicates with device in the Slave role.
  - **Slave Role**: Communicates with single device in Master Role.

In this Thesis, we are focusing mainly on the Advertising and Scanning phases of BLE devices. As shown in Figure 2.1, the network topology is a Broadcast topology, where there could be many advertisers and many scanners. In the iBeacon protocol however, we usually see the star topology, as there is one who advertises(beacon) and many scanners (users’ devices).
Chapter 2: Beacons and Indoor location-based services

Figure 2.1 BLE topology[10]

2.3 Beacons

Introduced in iOS 7, iBeacon is an exciting technology proposed by Apple Inc. to enable new location-aware information and services for mobile devices. Leveraging Bluetooth low-energy (BLE) for a short-range communication, a device with iBeacon technology can establish a region around itself. Each iBeacon (called a beacon) broadcasts a 20-byte unique ID, which is divided into three sections: proximity UUID (16 bytes), major number (2 bytes) and minor number (2 bytes). A beacon continuously broadcasts its unique ID via BLE to devices in its close proximity. The broadcast coverage of a beacon could be as small as 2 inches and as large as 230 feet away. Any mobile device entering such coverage of a beacon can receive its unique ID without a priori and explicit pairing procedures [11]. A related event will be triggered by the application of a mobile device when it enters or exists the covered region of the beacon, which can be applied to inform the device about its geographical location changes [12]. The coverages of different beacons overlap. Mobile devices in the overlapped area can pick up the unique IDs broadcasted by multiple beacons, and estimate their distances to those beacons by measuring the received signal strength (RSSI) [13]. The closer a device is to a beacon, the stronger the corresponding signal it can receive. Thus, the mobile device can obtain its relative location based on the knowledge of its estimated distances to each beacon. iBeacon-based indoor positioning systems have merits of both Internet of things and mobile internet and achieves the goal of indoor positioning [8]. It has the advantages of low power consumption, fast response, and accurate positioning. Beacons are usually small devices, consisting of a cover, a BLE chipset, a battery and a firmware for changing the settings of it. Figure 2.2 shows the parts a beacon consists of. Instead of other positioning systems, beacon is a 1-way transmitter to the receiving device. What is more, this device has to have a specific app installed on the device in order to trigger it and interact along with the beacon. The iBeacon technology
enables new ways of advertising and targeting the readers by paying attention to their physical location. More specifically, the technology enables what we call iBeacon Based Experiences - Engaging and informative experiences tied to their physical location instead of spamming the readers with intrusive ads. Even though iBeacon is originally for iOS,

![Figure 2.2 Beacon’s main parts](image)

Figure 2.2 Beacon’s main parts

SDK for Android has recently released to bring iBeacon technology to Android. The main advantage of iBeacon is that the application can run in the background and only shows up when another iBeacon is detected. This gives a good user experience. For the application developed in this thesis, we used iBeacons developed by Estimote.

### 2.4 Estimote

An Estimote is a beacon that broadcast radio signals based on Bluetooth Low Energy. Estimote is based on iBeacon technology and so are also thought to work with proximities. However, there are some methods in the API that allow the location of mobile devices and measure distances indoors, but the accuracy is still poor because of the changes in indoor environment. Estimote beacons Developer Kit contains three estimote beacons based on iBeacon concept. Estimotes follow iBeacon concept, that is part of the CoreLocation framework in iOS. iBeacon is similar to geofencing (virtual barriers or boundaries in some applications based on geopositioning) but uses BLE signal power instead of GPS to know one’s proximity to the beacon. It has the advantage of detecting proximity when the target is near the beacon instead of having to rely on a fixed GPS location. So, the proximity is relative to an identified beacon placed intentionally in one place.

Each beacon is advertising a packet of 30 bytes data. The most significant bytes of the packet concern:

- **UUID identifier**: is an identifier to distinguish the beacons of one company from others (16 bytes). For example, all the beacons belonging to a brand would have the
Chapter 2: Beacons and Indoor location-based services

same UUID. The Estimote’s UUID is B9 40 7F 30 F5 F8 46 6E AF F9 25 55 6B 57 FE 6D.

• Major: is also an identifier for a group of beacons (2 bytes). For example, all the beacons placed in a store from one company would have the same major. The beacons of other store from the same company shares the UUID but with different major. In this case, the major is 00 49.

• Minor: is used to distinguish the beacons from one group of beacons (2 bytes). For example, the beacons placed in a store share the same major and are identified by the minor number.

Figure 2.3 Estimote - iBeacon logo

The Estimote application allows you to configure these parameters, as well as change the:

• Advertising interval: is the period when an advertising packet is broadcasted. It can be configured from 50ms to 2000ms. This parameter is critical in battery life so it has to be configured according to the requirements of the application.

• Broadcasting power: is the physical power of the transmitted signal. Its values go from -30dBm to +4dBm. This parameter is also critical in battery life and has to be configured according to the requirements of the application.

iBeacon and Bluetooth Low Energy, are both good solutions for positioning that present the following advantages:

1. BLE is a good solution for an indoor environment, having a range up to 50 meters approximately and good accuracy.

2. Beacons are easily identifiable in an indoor environment (not only MAC Address), with its parameters.

3. iBeacon allows data transmission related to positioning and is easy to extract.
4. Estimote minimizes power consumption using Bluetooth Low Energy technology.

5. iBeacon is a new protocol presented by Apple and is earning the support of the industry.

6. The accuracy can be improved depending on the situation, by adjusting the advertising interval and power transmission.

7. Estimote is secure for user privacy because no connection or pairing is needed between the beacon and user device.

8. Beacons are also low-cost; the deployment is not as expensive as other technologies (e.g. IEEE 802.11).

2.5 Beacons and location-based services

Although the technology of iBeacon was introduced in the world at 2013, many companies are exploiting its potential and have created useful applications.

1. In Brazil, Nivea used a Bluetooth proximity beacon embedded in magazine ads, which parents could tear out from the page in the form of a wrist-band for their children. A smartphone app can be used to track how far the children walk away from their parents in places like beaches. Parents receive an alert when the child walks out of a pre-defined range, and can then bring them back to safety. Nivea is now seen as not just the maker of creams to be used in sunny locations, but as a brand which cares for its consumers’ children [14].

![Figure 2.4 Nivea’s campaign with iBeacons](image)

2. In Greece, iStorm, a premium shop retailer of Apple products, is the first company which introduced iBeacon technology. Whenever a person entered the shop, he just had to enable bluetooth and location services, and as long as he had the application of iStorm installed on his device, he could receive notifications of various products around him. What is more, he can check into the shop using his device in order to acquire loyalty points, which he can use in his next purchases.
Chapter 2: Beacons and Indoor location-based services

Figure 2.5 iStorm app utilizes iBeacon for increased customer experience

3. In UK, Virgin Atlantic, recognised as an innovator in the airline industry began trialling iBeacon in the Upper Class Wing at London’s Heathrow Airport in May 2014. Premium passengers of the airline received personalized notifications and offers via their iPhones. In the initial stages of the trial, customers were alerted to have their electronic boarding passes ready when they were close to a private security check, also passengers in the departures section of the airport were sent tailored offers like commission-free currency exchange deals. Virgin also used iBeacon technology to let passengers know about in-flight entertainment specials before they boarded the plane, and to let their staff know when temperatures dropped on outdoor decks (the beacons have in-built temperature sensors) of their airport lounges so they could give blankets to passengers [15].

Figure 2.6 Virgin Atlantic iBeacon implementation at the Heathrow airport

4. In USA, NFL installed iBeacons in MetLife Stadium in New York at 2014, which was hosting the Super Bowl as well as in Times Square. The alerts helped users find the closest entry gate, or where the restroom was in the stadium. Those in Times Square advised the people where to find their way around town or where they can grab their favorite team’s jersey. The alerts are mostly limited to practical news (like the nearest entry gate) or promoting in-store sales (say, for your favorite chocolate) in the first wave of establishments using it.
Chapter 3

Recommender Systems and Link Similarity Algorithms

We have 6.2 million customers; we should have 6.2 million stores. There should be the optimum store for each and every customer.

Jeff Bezos, CEO of Amazon.com

In the past few years, Internet has spread and continues to evolve rapidly. This development has resulted in the exponential increase in the amount of information. All this amount of info is available to all users who surf on the Internet. However, it requires a lot of time and effort to look through this enormous amount of information in order to collect, evaluate, process and finally choose whatever interests each user. Thus, a new need arouse. The need to create a system that would make this whole process for the user. This system would be able to guide him and direct him to identify and collect only the data suitable for him, depending on his wishes and needs. It is often necessary to make choices without sufficient personal experience, thus, we are looking for the opinion of other people, either by word of mouth or by reading articles on newspapers, recommendations about movies, books, general surveys or travel and restaurant guides.

3.1 Definition

A facility, or a Web application, that involves predicting user responses to options, is called a recommender system. Recommender systems use the opinions of a community of users to help individuals in that community more effectively identify content of interest from a potentially overwhelming set of choices [16]. Recommender systems play now an important role to the information overload problem, in which users are finding it difficult to locate the right information at the right time. We have seen many such systems, especially in the e-commerce section, in the form of virtual assistants. They combine user profiles, machine learning and information filtering and provide the users with a more intelligent and proactive information service [17]. They are applications which can be found in mostly e-commerce sites and they are suggesting interesting, useful, hot products, explaining why they recommend these products, and they provide consumers with
3.2 Recommendation Process

the necessary information to facilitate the decision making process. The research on recommender systems is not only motivated by the information overload, but by the lack of user knowledge in many domains, a cost-benefit tradeoff optimization and minimization of user interaction. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen, etc.[18]. The goal of a Recommender System is to generate meaningful recommendations to a collection of users for items or products that might interest them. Two good examples in order to pinpoint the exact use of such systems are:

- Offering news articles to on-line newspaper readers, based on readers preferences or reader interests.
- Offering customers of an on-line retailer suggestions about what they might like to buy, based on their past history of purchases, past ratings and/or product searches.

Recommender systems have become extremely common in recent years, and are applied in a variety of applications. The most popular ones are probably movies, music, news, books, research articles, search queries, social tags, and products in general. As we mentioned before, Recommender Systems are gaining widespread acceptance as a way of tackling the “information overload” problem. This problem affects our everyday experience while searching for information on a topic. To overcome this problem, we often rely on suggestions from others who have more experience on the topic. However, in the Web case where there are numerous suggestions, it is not easy to detect the trustworthy ones. The process of recommendation becomes controllable by shifting from individual to collective suggestions [19].

3.2 Recommendation Process

Every Recommender system follows a certain process in order to produce the suitable recommendations. Figure 3.1 shows exactly that process. As we can see, Mr A registers his preferences to the system’s database, either by rating a product or by buying it or either by looking on something.

Then, Mr A’s preferences are compared to other users, and based on the users who have similar preferences to him, the system proposes recommended items back to Mr A. The input data sources may be

1. User inputs concerning his interests, his needs(profile)
2. Item’s characteristics and profile
3. User’s interaction with the various items (ratings, buyouts, time remaining on a page)

The interaction between the user and the item may occur in an implicit or explicit manner. Implicitly is performed when a user rates an item. This rating may be a binary rating (buy, not buy), a symbol(thumbs up), a numeric scale (0-5 stars), oral, vocal, etc. In real world however, most users are not willing to rate an object. For example, when someone buys
something over the Internet, the item usually takes some days to be delivered. Until the
time the user gets the item and uses it, he may have forgotten that he has to rate it in the
store he bought it. Thus, many explicit ways of ratings have been developed. These ways
evaluate the user’s observed behaviour. For example, in an e-shop, the time a user stays
in an item page usually declares his interest for an item or not. Think of it as when a user
wants to buy an item, he usually checks item specifications, sees some item images, check
the other reviews. Even if, in the end, he does not buy the item, the user has showed a
rating for the specific item and the recommender system may suggest similar items.

![Figure 3.1 Recommendation process - adopted from [20]](image)

### 3.3 Recommender System Classifications

The first recommender systems were based on the usage of statistical analysis of inform-
ation (the number of times this information has been used), on historical analysis (the
most recent information is better than the older ones) or on content (the matching of item
characteristics or user’s info).

- The simplest approach that has emerged in the context of RS is the non-personalized.
  In these RS, the recommendations are the same for all users. They are selected either
  based on the popularity (item with most views, highest rating or most purchases) or
  they are based on products which the owners want to promote. Because of the non
  personalized character, the recommendations may not be as accurate as predicted.

- In most RSs, the recommendations are personalized and they are destined to a spe-
  cific user or they include specific items. The most important approaches in person-
  alized recommendations are

  1. Content-based filtering, which assumes that each user operates independently
     and this kind of RS exploits only information derived from item features.
2. Collaborative filtering algorithms, which recommend to target users those items that have been rated highly (by other users with similar preferences).

3. Hybrid algorithms, which attempt to combine the two previously mentioned approaches. This combination of content with rating data helps capture more effective correlations between users and items and this result in more accurate recommendations[19].

3.3.1 Content-based Filtering

In the content-based filtering approaches, the system recommends items to users taking into consideration the correlation between the item’s content and the users preferences. In particular, it exploits a set of attributes, which describes the items and recommends other items similar to those that exist in the user’s profile. The logic behind this approach is that a user who shows a particular preference for a set of items with the same content(keywords, characteristics, names), then it is quite possible that he may be interested in another item that belongs to the same category. As Meteren and Someren[29] mention, a content-based filtering system selects items based on the correlation between the content of the items and the user’s preferences as opposed to a collaborative filtering system that chooses items based on the correlation between people with similar preferences. These approaches usually are based on users purchase history or items he has rated and they do not take into account other users preferences. The similarity between items and users can be calculated offline and there is an explanation about the recommendation and why this item is being recommended, thus increasing the confidence between the users and the recommender system. The main advantage of these systems is that they overcome the cold start problem(new items can be recommended without having any ratings, since the information are collected from the item’s description). On the other hand, such a system cannot provide correct recommendations if there is not enough content information. What is more, new users may not receive suitable recommendations before they build their profile, as there is no diversity. That is, the user gets recommendations that are very familiar to her, since the recommended items are similar to those in her item profile.

3.3.2 Collaborative Filtering

In most CF approaches, only the item and users’ identifiers are accessible and no additional information over items or users is provided. Websites that provide recommendations in the form, “Customers who bought item i also bought item y”, typically fall under collaborative filtering approaches. The assumption which underlies these systems is that users who have agreed to something in the past tend to agree again in the future. GroupLens research group [21] introduced a collaborative filtering algorithm, known as user-based CF, because it employs users’ similarities for the formation of the neighbourhood of nearest users. Another CF approach is Item-based CF, which employs items’ similarities for the formation of the neighbourhood of the nearest users[22]. Pearson correlation and cosine similarity have mainly been used to calculate similarity in collaborative filtering algorithms. In particular, user-based CF algorithms use the first (Equation 3.1)
Chapter 3: Recommender Systems and Link Similarity algorithms

1, which measures the similarity between two users, \( u \) and \( v \). Item-based CF algorithms use a variation of adjusted cosine-similarity Equation 3.2)\(^2\), which measures the similarity between two items, \( i \) and \( j \). This method is more accurate as it normalises the bias from subjective rating.

\[
sim(u, v) = \frac{\sum_{i \in S} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in S} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in S} (r_{v,i} - \bar{r}_v)^2}}, \quad S = I_u \cap I_v.
\]  

Equation 3.1 takes into account only the set of items, \( S \), that are co-rated by both users. This, however, ignores the items rated by only one of the two users. The number of the latter items denotes how much their preferences differ. Especially for the case of sparse data, by ignoring these items we discard significant information. Analogous reasoning applies for Equation 3.2, which considers (in the numerator) only the set of users, \( T \), that both co-rated the examined pair of items. The same applies for WS, which is based on Equations 3.1 or 3.2. To address the problem, in the following, we will examine alternative definitions for \( S \) and \( T \).

CF is mentioned as the most widely adopted approach as it contains some characteristics which confront some content-based filtering weaknesses. It is very important that these systems look beyond the preferences of each user and can suggest new items to the user, thus creating needs and interests that they might not have previously. A pitfall of CF is the cold start problem: new items have received only few ratings, so they cannot be recommended; new users have performed only few transactions, so other users similar to them can be hardly found.

3.3.3 Hybrid Recommender Systems

Hybrid recommender systems combine two or more concepts from those presented, to increase performance. Combining different approaches limits weaknesses, as one covers the weaknesses of the other. The most common combination is the one that combines approaches to content-based and collaborative filtering approaches, thus tackling the problem of cold start for a new item.

As Burke denotes\([30]\) the mixed hybrid avoids the “new item” start-up problem, whereas, it does not get around the “new user” start-up problem, since both the content and collaborative methods need some data about user preferences to get off the ground.

---

\(^1\)Means \( \bar{r}_u, \bar{r}_v \) are the mean ratings of \( u \) and \( v \) over their co-rated items.

\(^2\)Means \( \bar{r}_u, \bar{r}_v \) are taken over all ratings of \( u \) and \( v \).
3.4 Recommender System Examples

There are many real-world recommender systems which can attract our attention. First and foremost, the Amazon was one of the first companies which applied RS on their website. Amazon.com extensively uses recommendation algorithms to personalize its Web site to each customer’s interests. Amazon.com uses recommendations as a targeted marketing tool in many email campaigns and on most of its Web sites’ pages, including the high traffic Amazon.com homepage. Clicking on the ‘Your Recommendations’ link leads customers to an area where they can filter their recommendations by product line and subject area, rate the recommended products, rate their previous purchases, and see why items are recommended. Amazon is very famous for its item-based CF Recommendation System.

Figure 3.2 Amazon recommendations

Another interesting case is Netflix. Netflix offered a prize of $1M to the first person or team to beat their own recommender algorithm, CineMatch, by 10%. There are many recommendation algorithms Netflix. People usually refer to the "rating prediction" algorithm that was researched in the Netflix Prize as the "Netflix Recommendation Algorithm", but that is by no means the only or the most important of the algorithms in the Netflix recommendation system. Netflix recommendations heavily depend on similarity. Similarity is an important source of personalization in our service. We think of similarity in a very
broad sense; it can be between movies or between members, and can be in multiple dimensions such as metadata, ratings, or viewing data. Furthermore, these similarities can be blended and used as features in other models. Netflix uses a complex hybrid RS.

![LinkedIn recommendations](Figure 3.4)

Another interesting website that has a recommendation system is LinkedIn, the famous professional network which provides news and feeds recommendations based on the persons you are connected with and the articles you have liked. It uses a CF recommendation engine to provide these recommendations. As we see in Figure 3.4, the articles which are presented are from the daily “Pulse” and the people that we follow or are connected to.

Another website is Youtube, where you can find recommendations based on the video you are currently watching, or recommendations based on the videos you have watched in the past. Both recommendation techniques use Content-based Filtering algorithms.

### 3.5 Sensor-based Recommendations

In our project, we tried something similar in terms of a conventional store. With the usage of beacons and a simple app, we keep a record of the amount of time a potential buyer spends in a store section. Thus, we can allocate an explicit preference for that store section of a user. As Table 3.1 depicts, in our project, a user can either visit a shop physically or browse our e-shop. Items are products which have been assigned to different beacons that have been installed in various sections of a store. There may be examples where a beacon is assigned to a single item, in order to maximize its sales. Finally, a user’s preference in our project is based on the time a user spends in a particular beacon point. Beacons have the ability, using an appropriate application, to capture the moment a user enters and exits their region (the first time the application gets the beacon’s signal and the last time it does), thus letting us know the amount of time s/he spends in a beacon region. Consequently, we are able to cluster time into time regions, and in the end we assign a rating, which expresses the preference of a user on an item.

The data set used in a recommender system is always represented by a matrix that is known as the utility matrix. A simple example of an RS in a conventional shop is illustrated...
3.5 Sensor-based Recommendations

<table>
<thead>
<tr>
<th>User</th>
<th>E-shop RS</th>
<th>Conventional Shop Rs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Customer surfing on the online system</td>
<td>Customers inside the shop</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preference</td>
</tr>
<tr>
<td>Items on the database</td>
</tr>
<tr>
<td>HighRatings given by users to items</td>
</tr>
</tbody>
</table>

Table 3.1 Term Definition

in Table 3.2. Users are represented by their user IDs, such as user with ID 4 in the leftmost column, whereas the items are represented by the beacon major number. The preference values in our example are the actual number of seconds a user spends in a store section. Notice that most values in the preference field are left blank, which indicates that the users have never been to the corresponding sections. In reality, since a row in the utility matrix will include all the preferences to each store in the shop, the matrix will be much more sparse than that in the example, with the typical user only to visit a tiny fraction of all the available sections.

<table>
<thead>
<tr>
<th>User ID (Apple)</th>
<th>54132</th>
<th>11353 (Samsung)</th>
<th>86766 (TV Section)</th>
<th>33377 (Camera)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>122 sec</td>
<td>255 sec</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>600 sec</td>
<td>-</td>
</tr>
<tr>
<td>234</td>
<td>50 sec</td>
<td>80 sec</td>
<td>-</td>
<td>200 sec</td>
</tr>
</tbody>
</table>

Table 3.2 Mapping preference between user and item

For example, user with ID 234 passes from three different sections in a shop where beacons have been installed. Specifically, his app detects the signal of beacon with major:54132(Apple section) for 50 seconds, whereas the user stays in the Samsung section for 80 seconds and in the camera section for 200 seconds. We can conclude that this specific potential buyer shows more interest for cameras than for mobile phones. The recommendation process, especially for collaborative filtering, which we will explain later, consists of three stages:

1. Stage 1: Formation of the user or item neighbourhood with objects of similar ratings and behaviour.

2. Stage 2: Generation of a top-N list with algorithms that construct a list of top-N item recommendations for a user.


Based on the aforementioned work on Recommender Systems, we wanted to build one such system which will be based on real time recommendations. What is more, we want
Chapter 3: Recommender Systems and Link Similarity algorithms

to install iBeacons inside a store in order to know exactly the position of the customer, thus being able to provide him location-based recommendations. Our goal is to be able to predict customers needs and analyse real time correlations between customers and items. A common scenario for accomplishing our recommender system is as follows:

A store contains 4 beacons which are assigned to 4 different sections. The first one, B1, is installed in the entrance to keep track of users’ entrance and exit, and to provide him with general discounts for shop’s deals. The other three (B2, B3, B4) are assigned to jeans section, shirts section and shoes section, correspondingly. Let’s assume there are two users, visiting this store, having the application installed on their devices. As shown in Figure 3.5(a), user $U_1$ visits jeans section when he enters the store. Afterwards, he goes to the other two sections. In all three sections, he stays for more than 3 minutes, which is assumed as a positive preference rating to the products that belong in these sections. Next, another user, $U_2$, enters the store. First he visits the shirts section and then proceeds to the shoes section. If this user’s behaviour results in a positive rating for the products of shoes and Jeans sections (by staying there for a time period greater than a predefined threshold), then when he decides to leave the second section, a notification appears on his device prompting him to check the app’s recommended items for him, as shown in Figure 3.5(b). This recommendation may include the Jeans section and any product deals that exist in it (We recommend you to visit Jeans Section because $U_1$ visited it in the past and you both have visited Jeans and Shoes sections). Thus, you can easily understand how our Recommender System would be exploited in order to provide the best services to customers.

![Figure 3.5](image-url)  
(a) Recommendation Scenario  
(b) Recommendation Scenario

**Figure 3.5** Context-aware recommendations based on beacons - Example
Chapter 4

Link Similarity algorithms

As we saw in Chapter 3, in a Collaborative Filtering recommender system, it is very important to create a proper neighbourhood in order to select the appropriate nearest neighbours. Then, we want to retrieve their items, which have been rated similarly by the target user and provide the top-N list of item recommendations to him. More generally, a similarity measure can be used to cluster objects. For example, collaborative filtering algorithm in a recommender system finds similar users/items and groups them based on their preferences. This problem can be represented as a graph problem, where the vertices of graphs may represent users and items. Preference of a user towards an item may be represented by an edge. This simple graph, which has only two kind of vertices(entities), users and items, and its edges are directed from one level (users) to the other level (items) is called bipartite graph.

4.1 Introduction to graphs

Graphs offer a convenient way to represent various kinds of mathematical objects. Essentially, any graph is made up of two sets, a set of vertices and a set of edges. Depending on the particular situation we are trying to represent, however, we may wish to impose restrictions on the type of edges we allow. For some problems we will want the edges to be directed from one vertex to another; whereas, in others the edges are undirected. A simple graph \((V, E)\) consists of a nonempty set representing vertices, \(V\), and a set of unordered pairs of elements of \(V\) representing edges, \(E\). A simple graph has:

- no arrows,
- no loops, and
- cannot have multiple edges joining vertices.

As we mentioned above, in recommender systems, we usually use bipartite graphs. A bipartite graph is a graph whose vertex-set can be split into two sets in such a way that each edge of the graph joins a vertex in first set to a vertex in second set. In other words, bipartite is a special graph of \(k\)-partite graph where \(k = 2\). In the same way as with the bipartite graphs, if we can divide the vertex set into three disjoint non empty sets \(V1, V2\) and \(V3\) so that vertices in the same set are not adjacent we get a tripartite graph. In Figure 4.1 we can see an example of a unipartite graph. In this example, if we remove the edge between nodes \(T_3\) and \(T_4\), we have a tripartite graph.
4.2 SimRank

**Example 1** Figure 4.1 shows an example of a tripartite graph. As shown, there is one user, three movies and four movie genres (tags). Let’s assume that we want to compare the similarities among movies. User $U_1$ has rated positively (rating $\geq 3$) movies $M_1$, $M_2$, $M_3$. Movies $M_1$ and $M_2$ are linked to tag $T_1$. This means that both movies ($M_1$ and $M_2$) belong to the same movie genre. Please, also notice that movies $M_2$ and $M_3$ are linked to tag $T_4$. Moreover, notice that tags $T_3$ and $T_4$ are connected, which means that they are highly correlated. In other words, they characterise commonly the same movies and thus there is a reciprocal link that connects them.

**Figure 4.1** A visual representation of our toy example (User-Movie-Tags)

Based on the aforementioned tripartite graph, the user-movie bipartite graph denotes the long term preferences of a user, whereas the movie-tag bipartite graph denotes the movie genre a movie belongs to.

**4.2 SimRank**

In 2002, Jeh and Widom [5] proposed a complementary approach in the problem of measuring similarity, applicable in any domain with entity-to-entity relationships. This measure is based on the idea that “two entities are similar if they are related to similar entities”.

Chapter 4: Link Similarity algorithms

This general similarity measure is known as SimRank and it is based on a simple and intuitive graph-theoretic model.

\[
s(a, b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} s(I_i(a), I_j(b))
\]

(4.1)

In the above Equation, \( s(a, a) = 1 \) and \( 0 < C < 1 \). \( C \) represents the degree of attenuation in similarity propagation. The above graph may be iterative but it converges fast, in most situations usually after the fourth or fifth iteration. In order to reduce its complexity, we set a relatively high threshold for convergence.

SimRank model for bipartite user-item network consists of two type of objects. Here, similarity of items and similarity of people are mutually-reinforcing notions:

- People are similar if they purchase similar items.
- Items are similar if they are purchased by similar people.

In Equation 4.2 and 4.3, we can see how the SimRank algorithm transforms for bipartite graphs. Let \( s(A, B) \) denote the similarity between persons \( A \) and \( B \), and let \( s(c, d) \) denote the similarity between items \( c \) and \( d \).

\[
s(A, B) = \frac{C}{|O(a)||O(b)|} \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} s(O_i(a), O_j(b))
\]

(4.2)

\[
s(c, d) = \frac{C}{|I(c)||I(d)|} \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} s(I_i(c), I_j(d))
\]

(4.3)

Next, we apply the SimRank algorithm in our running Example 1, as shown in Figure 4.1. Let’s assume that we want to compute the similarity between movies. Table 4.1 shows the similarity score between movies \( M_1, M_2, \) and \( M_3 \), based on Equation 4.3. As shown, the similarity score is the same for all movie pairs, if we take under consideration only the in-links of our graph. Please notice that we cannot distinguish well the movies’ similarities, since all values are 0.8, as shown in Table 4.1. In Equation 4.4 we compare the similarity between Movie \( M_1, M_2 \) and \( M_3 \). SimRank converges from the first iteration and the same value can also be seen at Table 4.1

\[
s(M_1, M_2) = \frac{C}{|I(1)||I(1)|} \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} s(I_i(1), I_j(2))
\]

(4.4)

\[
= \frac{0.8}{1*1} \sum_{i=1}^{1} \sum_{j=1}^{1} s(U_i, U_j) = 0.8 * 1 = 0.8
\]

Please notice that the high algorithmic complexity of this method makes it necessary to apply optimisation techniques for its computation, because, as the number of edges increases, this method becomes impractical to use.
4.3 P-Rank

In 2009, Zhao et al. [4] proposed P-Rank (Penetrating Rank), an enriched version of SimRank. The authors wanted to test "how similar two entities are within an information network". With P-Rank, they could effectively compute the structural similarities of entities in such networks. The difference between SimRank and P-Rank was that the latter was taking into consideration both in and out-link relationships, whereas the first was calculating the similarity based only on in-link edges. The two-fold meaning of P-Rank is elaborated as follows:

1. Two entities are similar if they are referenced by similar entities.
2. Two entities are similar if they reference similar entities.

### Table 4.1 Structural Similarity Scores for SimRank algorithm

<table>
<thead>
<tr>
<th></th>
<th>SimRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s(M_1 - M_2)$</td>
<td>0.8</td>
</tr>
<tr>
<td>$s(M_1 - M_3)$</td>
<td>0.8</td>
</tr>
<tr>
<td>$s(M_2 - M_3)$</td>
<td>0.8</td>
</tr>
</tbody>
</table>

A lot of newer researches have been made in order to compute SimRank results faster. In 2008, Antonellis et al. [23] argued that SimRank fails to properly identify query similarities on a historical click graph. They also presented an enhanced version of SimRank, which assigns weights to each edge. They named their method SimRank++ and their tested results on actual click graphs and queries from Yahoo! yielded more and better query rewrites.

In 2010, Li et al. [20] argued that the iterative method can be infeasible when the real world networks change frequently and contain many nodes. Thus, they suggested a non-iterative method that enabled users to update the similarity scores incrementally and to derive similarity for an arbitrary subset of nodes. They rewrote SimRank equation into a non-iterative form by using Kronecker product and vectorization operators, as shown in Equation 4.5. This Equation contains vec operator, $vec(S)$, which is the vectorization of the matrix $S$ (similarity matrix which contains the similarity scores between each nodes pair) in a linear transformation which converts the matrix into a column vector, the dumping factor $c$, the normalized adjacency matrix $\tilde{W}$ and the vec operator of identity matrix, $vec(I)$.

$$vec(S) = (1 - c)(I - c(\tilde{W} \bigotimes \tilde{W}))^{-1}vec(I)$$  \hspace{1cm} (4.5)

In our project, we created a version for calculating similarities based on this non-iterative form, but we will not give more details on this method.

Two further research papers, from Yu et al. [24] and Tao et al. [25] have shown us that the improvement of SimRank method can be achieved, in terms of efficiency.

### 4.3 P-Rank

In 2009, Zhao et al. [4] proposed P-Rank (Penetrating Rank), an enriched version of SimRank. The authors wanted to test "how similar two entities are within an information network". With P-Rank, they could effectively compute the structural similarities of entities in such networks. The difference between SimRank and P-Rank was that the latter was taking into consideration both in and out-link relationships, whereas the first was calculating the similarity based only on in-link edges. The two-fold meaning of P-Rank is elaborated as follows:

1. Two entities are similar if they are referenced by similar entities.
2. Two entities are similar if they reference similar entities.
In Equation 4.6, the relative weight of in- and out-link directions is balanced by a parameter $0 < \lambda < 1$. We can change the value of $\lambda$ to give more or less importance to similarity flows from in-links or out-links. Thus, if we set $\lambda = 1$, P-Rank becomes equal to SimRank. The solution to the P-Rank formula can be reached by computing its iterative form to a fixed point.

$$s(a, b) = \lambda \times \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{I(a)} \sum_{j=1}^{I(b)} s(I_i(a), I_j(b))$$

$$+ (1 - \lambda) \times \frac{C}{|O(a)||O(b)|} \sum_{i=1}^{O(a)} \sum_{j=1}^{O(b)} s(O_i(a), O_j(b))$$

(4.6)

For our running Example 1, we show in Equation 4.7 how we compute a similarity value based on P-Rank algorithm (see Equation 4.6). At the end of the Equation, you can see better how this iterative algorithm works, as in order to compute the similarity value of one type (i.e. movies), we have to compute the similarity value of another type (i.e. tags) and so on.

$$s(M_1, M_2) = 0.5 \times \frac{0.8}{|I(M_1)||I(M_2)|} \sum_{i=1}^{I(M_1)} \sum_{j=1}^{I(M_2)} s(I_i(M_1), I_j(M_2))$$

$$+ (1 - 0.5 \times \frac{0.8}{|O(M_1)||O(M_2)|} \sum_{i=1}^{O(M_1)} \sum_{j=1}^{O(M_2)} s(O_i(M_1), O_j(M_2))$$

(4.7)

$$= 0.5 \times \frac{0.8}{1 \times 1} \sum_{i=1}^{1} \sum_{j=1}^{1} s(O_i(M_1), O_i(M_1)) + (1 - 0.5 \times \frac{0.8}{1 \times 3} \sum_{i=1}^{1} \sum_{j=1}^{3} s(O_i(M_1), O_j(M_2))$$

$$= 0.4 \times s(U_1, U_1) + (1 - 0.5 \times \frac{0.8}{1 \times 3} (s(T_1, T_1) + s(T_1, T_3) + s(T_1, T_4))$$

$$= 0.4 \times 1 + (1 - \frac{0.4}{3} (1 + s(T_1, T_3) + s(T_1, T_4))$$

In our running Example 1, we can see in Table 4.2 that the similarity of the three movies is not the same any more.

<table>
<thead>
<tr>
<th></th>
<th>SimRank</th>
<th>P-Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s(M_1 - M_2)$</td>
<td>0.8</td>
<td>0.62</td>
</tr>
<tr>
<td>$s(M_1 - M_3)$</td>
<td>0.8</td>
<td>0.51</td>
</tr>
<tr>
<td>$s(M_2 - M_3)$</td>
<td>0.8</td>
<td>0.62</td>
</tr>
</tbody>
</table>

Table 4.2 Structural Similarity Scores

As shown in Table 4.2, the similarity between $M_1 - M_2$ and between $M_2 - M_3$ is greater than the similarity between $M_1 - M_3$ if we take into account both the in-links and the out-links of the graph. As can be seen in Table 4.2, if we had to propose the most similar movie to $M_1$, that would be the $M_2$ based on P-Rank. From the other hand, based on SimRank,
4.3 P-Rank

we would suggest all movies as similar to $M_1$, as shown in Table 4.1. We need to mention that the out-links of the graph of our running Example 1 (see Figure 4.1) add valuable information to the link prediction problem, information which we lose if we use SimRank algorithm.

Algorithm 1 illustrates the iterative procedure for computing P-Rank in a graph denoted by $G$. Let $n$ be the number of vertices in $G$ and $k$ be the number of iterations executed until P-Rank converges to its fixed point. For every vertex pair $(a, b)$, an entry $R(a, b)$ maintains the intermediate P-Rank score of $(a, b)$ during iterative computation. Because the $(k + 1)$-th iterative P-Rank score is computed based on P-Rank scores in the $k$-th iteration, an auxiliary data structure $R^*(a, b)$ is maintained accordingly.

Because $G$ can be very large, which makes it difficult to be stored in main memory, any advanced data structures that optimises external memory accesses can be applied.

Algorithm 1 first initializes $R(a, b)$ (Lines 1-4) We want to mention here that if two vertices are the same, the score for these vertices are 1, whereas if they are different, the score is 0. So every computation of this algorithm does not contain any calculations, but rather comes to the point where we decide if two vertices are the same or not and assign a value of
1 or 0. During iterative computation, P-Rank’s $R^*(a, b)$ (the score of each pair) is updated by $R(a, b)$ in the $k$-th iteration (Lines 6 - 18). We can see that we use the P-Rank Equation 4.6 in Lines 13 and 18. In these Lines, we compute the similarity value that occurs from each iteration of the algorithm. For two specific nodes, $a$ and $b$, we compute both the in-links and the out-links and then we add these two values to the similarity score of the pair $a, b$. As we can see in Algorithm 1, we calculate a similarity score for both in-links and out-links. Then $R(a, b)$ is substituted by $R^*(a, b)$ for further iterations (Lines 19 - 21). The iterative procedure stabilises rapidly and converges to a fixed point within small numbers of iterations. Finally, we get the similarity score for every possible vertex pair of the graph (Line 22).

We want to mention that the algorithm calculates the similarity score for all possible nodes $(a, b)$. P-Rank takes into account of both in- and out-link relationships of entity pairs and penetrates the structural similarity computation beyond neighbourhood of vertices to the entire graph.

The advantages of P-Rank are its semantic completeness, robustness and flexibility under different graph settings. The results confirm the applicability and comprehensiveness of P-Rank, as well as its significant improvement over SimRank.

4.4 OmniRank

Based on the previous two similarity measures over a structural network, we derived an extension of them and we called it OmniRank (Omniscient Rank, where omni refers to multi direction). This similarity extends the bipartite character of the SimRank algorithm to tripartite graphs.

Our work is inspired by the work of Xiang et al.[4]. To incorporate the time dimension into their models, Xiang et al.[4] proposed the construction of tripartite graphs (i.e., users, products, sessions) known as Session-based Temporal Graph (STG). Notably, STG does not incorporate edges among nodes of the same set, thus, failing to exploit information from all three unipartite networks (user-user, product-product and session-session). For instance, STG does not have links among user or session nodes. However, intuitively friends tend to buy similar products at close time points, which means that friendship links could leverage the accuracy of product recommendations. As we mentioned before, in a recommender system, when there is no data, people tend to listen to other people opinions through word of mouth, especially if they have something in common (friendship). Based on that we believe there can be more info in a tripartite graph containing parallel nodes showing friendship between people or the same store section that items are found or same sections that people tend to buy particular items. What we achieved by creating Omni-rank:

- We propose the construction of the tripartite graph, consisting of 3-disjoint sets of nodes (i.e. users, products, tags), and incorporates edges among nodes of the same set, including also three unipartite graphs, i.e. user friendship network, tag-tag network and product-product network.

- We compared our method with other state-of-the-art algorithms on two real datasets.
It will be shown that our OmniRank algorithm prevails its predecessors and achieves an significant improvement in terms of effectiveness/accuracy for product recommendation against all its competitors (see Section 6.4).

In Equation 4.8, the relative weight of in-link, out-link, parallel in-link and parallel out-link directions is balanced by four parameters, $w, x, y$ and $z$ where $w + x + y + z = 1$. We can change the values the parameters to give more or less importance to similarity flows from in-links out-links, parallel in-links or parallel out-links. Again, if we set $y$ and $z = 0$, OmniRank becomes equal to P-Rank and if the $x$ value becomes 0, OmniRank becomes equal to SimRank. The solution to the OmniRank formula can be reached by computing its iterative form to a fixed point.

$$s(a, b) = w \times \frac{C}{|TI(a)||TI(b)|} \sum_{i=1}^{|TI(a)|} \sum_{j=1}^{|TI(b)|} s(TI_i(a), TI_j(b)) + x \times \frac{C}{|TO(a)||TO(b)|} \sum_{i=1}^{|TO(a)|} \sum_{j=1}^{|TO(b)|} s(TO_i(a), TO_j(b)) + y \times \frac{C}{|UI(a)||UI(b)|} \sum_{i=1}^{|UI(a)|} \sum_{j=1}^{|UI(b)|} s(UI_i(a), UI_j(b)) + z \times \frac{C}{|UO(a)||UO(b)|} \sum_{i=1}^{|UO(a)|} \sum_{j=1}^{|UO(b)|} s(UO_i(a), UO_j(b))$$

As we have seen in our running Example 1, if we take the parallel links into consideration as simple links, they do not add any more information to our problem and the similarity scores between two nodes are the same. Based again in our running Example 1, we compute the similarity values between two movies. The better results of our method is due to the fact that the parallel link between $T_3$ and $T_4$ brings even closer the pair $(M_2, M_3)$ than the pair $(M_1, M_2)$ because of the reciprocal link between $(T_3, T_4)$. Thus, our intuition is confirmed since the similarity $s(M_2 - M_3)$ is higher than the similarity $s(M_1 - M_2)$. That is, our OmniRank method can capture better the similarity between $M_2$ and $M_3$ through the similar tags ($T_3$ and $T_4$). As can be seen in Table 4.3, OmniRank outperforms all other methods in Example 1. In other words, we come to the conclusion that the reciprocal link between the two nodes $(T_3, T_4)$ adds valuable information to the link prediction problem of Example 1. OmniRank calculates different scores for the three different pairs, thus separating the nodes similarity even better and achieving greater accuracy when recommending objects.

<table>
<thead>
<tr>
<th></th>
<th>SimRank</th>
<th>P-Rank</th>
<th>OmniRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s(M_1 - M_2)$</td>
<td>0.8</td>
<td>0.62</td>
<td>0.472</td>
</tr>
<tr>
<td>$s(M_1 - M_3)$</td>
<td>0.8</td>
<td>0.51</td>
<td>0.379</td>
</tr>
<tr>
<td>$s(M_2 - M_3)$</td>
<td>0.8</td>
<td>0.62</td>
<td>0.476</td>
</tr>
</tbody>
</table>

Table 4.3 Structural Similarity Scores
Algorithm 2: OmniRank($G, w, x, y, z, C, k$)

**Input:** A graph $G$, the relative weight factors $w, x, y, z$, the damping factor $C$, the iteration number $k$

**Output:** OmniRank score $s(a, b)$, $\forall a, b \in G$

1. **foreach** $a \in G$ **do** /* Initialization */
   2. **foreach** $b \in G$ **do**
      3. if $a == b$ then $R(a, b) = 1$
      4. else $R(a, b) = 0$
   5. while ($k > 0$) **do** /* Iteration */
      6. $k \leftarrow k - 1$
      7. **foreach** $a \in G$ **do**
         8. **foreach** $b \in G$ **do**
            9. $in \leftarrow 0$
            10. **foreach** $i_a \in I(a)$ **do**
                11. **foreach** $i_b \in I(b)$ **do**
                    12. $in \leftarrow in + R(i_a, i_b)$
                    13. $R^*(a, b) \leftarrow w \times \frac{C \times in}{|I(a)||I(b)|}$
                    14. $out \leftarrow 0$
            15. **foreach** $o_a \in O(a)$ **do**
                16. **foreach** $o_b \in O(b)$ **do**
                    17. $out \leftarrow out + R(o_a, o_b)$
                    18. $R^*(a, b) += x \times \frac{C \times out}{|O(a)||O(b)|}$
                    19. $pIn \leftarrow 0$
            20. **foreach** $p_i_a \in PI(a)$ **do**
                21. **foreach** $p_i_b \in PI(b)$ **do**
                    22. $pIn \leftarrow pIn + R(p_i_a, p_i_b)$
                    23. $R^*(a, b) \leftarrow y \times \frac{C \times pIn}{|PI(a)||PI(b)|}$
                    24. $pOut \leftarrow 0$
            25. **foreach** $p_o_a \in PO(a)$ **do**
                26. **foreach** $p_o_b \in PO(b)$ **do**
                    27. $pOut \leftarrow pOut + R(p_o_a, p_o_b)$
                    28. $R^*(a, b) += z \times \frac{C \times pOut}{|PO(a)||PO(b)|}$
            29. **foreach** $a \in G$ **do** /* Update */
                30. **foreach** $b \in G$ **do**
                    31. $R(a, b) = R^*(a, b)$
            32. return $R^*(a, b)$

Our Algorithm is presented in pseudocode form, in Algorithm 2, and it is an extension of Algorithm 1. The difference is that we introduce four variables, $w, x, y, z$, where $w + x + y + z = 1$. First of all, it is essential to tune the variables of our method. In order to solve different link prediction problems, we assign appropriate values to the variables.

Algorithm 2 first initializes $R(a, b)$ (Lines 1-4), the same as P-Rank. During iterative computation, OmniRank’s algorithm is the same as P-Rank algorithm (Lines 6 - 18). Next,
Omnirank continues to update $R^*(a, b)$ by $R(a, b)$ in the $k$-th iteration. As we mentioned before, Algorithm 2 also takes into consideration the parallel links of a graph (Lines 19 - 28). The process is the same as calculating all other edges of a graph. Then $R(a, b)$ is substituted by $R^*(a, b)$ for further iteration (Lines 29 - 32). The computation of the similarity value between a node pair $(a, b)$ is calculated in Lines 13, 18, 23 and 28. The iterative procedure stabilizes rapidly and converges to a fixed point within small numbers of iterations. Finally, we get the similarity score between each possible vertex pair of the graph, $G$. Thus, we can check which pair is more close than the others and sort all possible pairs in a more similar order.
Beacommender System is a service-oriented web application which provides the recommendation service to everyone who requests it. In Figure 5.1 you can see the architecture of Beacommender. Beacommender System consists of a website, hosted in "beacommender.eu", a mobile application for iOS devices, a JAVA servlet, the web-server of our system, and the database of it which is stored in a hosting site and uses MySQL technology.

![Beacommender System architecture](image)

**Figure 5.1** Beacommender System architecture

The website of our system is an e-shop of a virtual store. This e-shop offers various electronic devices to the potential buyers. The website offers an interface to the users so as to be able to interact with our recommendation engine leading to appropriate recommendations to each user. In order a user to receive better recommendations, there is the option of registering to the website. All users’ ratings are stored in our web database. The mobile application can be used inside the conventional store in order to receive special discount deals based on their location inside the store as well as to keep track of person’s spending time in each different store. Moreover, the same recommendation engine works in the mobile application. Please, bear in mind that each user requires only one account which can be used to both the e-shop and the mobile application.
5.1 Website

As we mentioned before, the website is an e-shop of a virtual store. This store contains mostly electronic devices (mobile phones, cameras, PC’s). In the website, the user must create a personal account to be able to rate products. However, in order a user to get recommendations based on a product he sees, it is not necessary for a user to be logged-in. The website was created based on OpenCart CMS (Content Management System). OpenCart is based on MVC (Model - View - Controller) architecture, in which there are different php files serving as controller’s, other php files serving as the CMS’s model and “.tpl” files containing HTML and javascript code which act as the view Files. We have kept the default theme’s design. We have injected code blocks in some of the files, properly enabling the website to communicate with the web service and get back the appropriate recommendations. Finally, we have created some custom pages which are useful to the website’s administrators.

5.1.1 Homepage

In the website’s homepage, as you can see in Figure 5.2, you can view some featured products of the website as well as register or login from the top navigation menu.

![Beacommender.eu Homepage](image)

**Figure 5.2** Beacommender.eu Homepage
Chapter 5: Beacommender System

5.1.2 Product’s Page

The most functional page of our website is an product’s page. Whenever you click on an product, a page for that product will open. This page contains the product’s photos, its price, the button which enables you to add it to your cart, a short description of the product as well as any characteristics of it the website owner has added.

There is also the option of viewing the product’s ratings, as shown in Figure 5.3, as well as the ratings date. What is more, if you have logged in to the website, you can rate a product yourself.

![Figure 5.3 Products’ ratings](image)

After that, when we go to the bottom of the page, some interesting proposals may arise. First of all, the first recommendations, that may come up, are the related products, which are assigned manually by the e-shop administrator. These products may be related somehow (having the same manufacturer, or belonging to the same product category). As we mentioned before, these relations are defined by the administrator. An example can be seen in Figure 5.4.

Below the related products, someone can find the recommendations produced by the recommendation engine. As we can see in Figure 5.5, there are two different recommendations categories. These categories are based both on Collaborative Filtering and the User-based and Item-based CF algorithms, mentioned in Section 3.3.2.

The first type of recommendations derives from User-based collaborative filtering. We have used two different algorithms for similarity, Pearson Correlation and Jaccard Similarity. Both are very fast algorithms and the recommendations occur instantly. The User-based CF recommendations can be found under the lexicon “Related Products based on Users’ Ratings”. We will explain later in Section 5.2 how these recommendations are produced, based on the database’s entries.

In our project, we get the top - 4 most similar neighbours to the target user. From these four neighbours, we count all their rated items and we propose those items which have been positively rated more times in the neighbour of the customer. We also apply a mini-
5.1 Website

**Figure 5.4** Related Products

**Figure 5.5** Recommended Products
mum rating threshold of 2, meaning that the candidates for nearest neighbours must have rated more than 2 movies.

The second type of recommendations derives from Item-based collaborative filtering. In this kind of recommendation, it is not necessary for a customer to be logged in, in order to receive recommendations. Those are the recommendations that are below the lexicon “Customers who bought this product also bought”. In this kind of recommendation, we used as similarity measure both cosine similarity and Omnirank.

As the number of products and users were not very large, the response time when using Omnirank for calculating similarity was quite short. The item-based CF algorithms calculates the top-4 most similar items, the top - 4 most similar neighbours as we discussed earlier in the user-based CF. However, in the item-based CF the most similart neighbours are also the items that would be proposed to the customer.

Finally, the last thing we want to mention is that we have also inserted a function in the administration panel of the website. We wanted the administration of the site to have a notion of how the users respond to the recommendations, so, as seen in Figure 5.6, we have added a table which presents all the items that have been recommended. Furthermore, the administrator can view the times a user has clicked on a recommendation and finally the times a user has added a recommended product to his cart as well as a purchase that has resulted from a recommendation. These analytics play an important role to the success of the recommendation, as, from this screen, the administrators of a shop may understand the value recommendation engine provides to their sites.

**Figure 5.6** Recommendation Results page

### 5.2 Web Database

The project’s database uses MySQL and is stored with the name ‘beaconrec’. MySQL is an open-source relational database management system (RDBMS); in July 2013, it was the world’s second most widely used RDBMS, and the most widely used open-source client-server model RDBMS. MySQL is a popular choice of database for use in web applications.
5.2 Web Database

It uses SQL, it is very fast, reliable and easy to use, it runs on a server and, above all, it is free to download and easy to use. This database contains tables for both the website and all the other necessary functions required for our web service to be in full functionality.

The website contains many tables which refer to the lot of functionalities the website has. In total, our database contains 134 tables, as you can see in Figure 5.7. For our recommendation model, we just use a simple table.

![Database Table Diagram]

**Figure 5.7 Tables of database**

**Rs_review table contains:**

- a review_id column, which is the unique key in this table,
- a product_id column which is a foreign key to the table rs_product and shows the item that has received a rating,
- a customer_id field which is a foreign key to the table rs_customer and relates this field with each unique customer that has added a rating for a product
- rating field which depicts the rating value the customer has assigned to a product
- the date_added field which shows the timestamp of the entry.

We store website ratings in a different page than those ratings coming from mobile application and corresponding to different store sections. For mobile ratings, which are stored based on the time each user stays inside the beacon range, we use the rs_ios table. Please, notice that although we said that beacon range reaches even 50m, we only store a preference for proximity that is less than 3m.

The rs_ios table contains:

- a record_id column which is the unique key in this table
- a customer_id column which is a foreign key and points to the rs_customer table, showing which customer was at the section and made the entry to the database
• a beacon_major column which shows to which store section the customer was, as each beacon’s major is assigned to a section

• a time column, which shows how long the customer has stayed in this section

• a rating column, which goes along with the time. We have concluded that time stayed in each section is assigned to a rating based on the following:
  – Rating : 1 if time <= 40 sec
  – Rating : 2 if 40 sec < time <= 70 sec
  – Rating : 3 if 70 sec < time <= 100 sec
  – Rating : 4 if 100 sec < time <= 200 sec
  – Rating : 5 if time > 200 sec

• a time_occurred column which shows the timestamp of the event

5.2.1 User-based CF recommendation example

Let’s explain now an example of our collaborative filtering algorithm. In Figure 5.5, we have seen that a user, who has made a successful login to the website, receives as recommendations via user-based CF four different products (Apple Cinema 30", HTC Touch HD, MacBook Air and MacBook Pro). This user has as customer_id the number 1. In Figure A.1, which can be found in Appendix A, we can see that customer_id 1 has made two positive ratings (>3) for the products with ID’s 43 and 41. At this moment, we don’t need to know to which product these ID’s are related to. Let’s see how similar this user is to other users. Customer_id 2 has made a positive rating to product_id 40, which however is negative rated by user 1, so customer_id 2 is not taken into consideration. Customer_id 3 has made two positive ratings, one at product_id 40 and one at 42. The first is negatively rated by customer_id 1 and the other is not rated at all. Next, we see the same results with customer_id 4, 5 and 6. As we can see, customer_id 1002 has rated positively products with id’s 40, 41, 42, 43, 45 and 28. Now, customers with ID 1 and 1002 have two common ratings, while customer with ID 1002 has rated different products than those customer 1 has, so he can be used from our method to recommend some products to customer 1. As we can see, products 41 and 43 are co-rated, so they will not be recommended. What is more, product 40 which is recommended by customer 1 cannot be proposed as well. So, 4 options are left. Products with ID 42, 44, 45 and 28. As you can see in Figure 5.8, product_id’s 44, 45 and 28 correspond to the item’s shown that are recommended before, whereas product with ID 42 is Apple Cinema 30”.

5.3 Web Server

MySQL databases are only accessible from certain IP’s, for security reasons. The website is hosted on the same server the database is hosted, so in that case we have no problem as to access the database. But as we wanted to use the same database for providing recommendations to websites and mobile devices, we had to use a web server in order to do
all the background job and just push the required data to the front-ends. Thus, we created a java Servlet, which is hosted on a static IP over the Internet and is allowed to access the database in order to fetch the data we want from it. Moreover, this servlet runs all the complex algorithms of Collaborative Filtering, which provide us the recommendations we want. In our project, we used Eclipse as Java working environment, and the JDBC API, which defines how a client may access a database. Based on this API, we could have access to the "Beaconrec" database and we could get all the data we wanted for running our project’s functions. In simplest terms, JDBC driver makes it possible to do three things:

1. Establish a connection with a data source
2. Send queries and update statements to the data source
3. Process the results

In Appendix B you can see how you can achieve a connection using JDBC to a database. Furthermore, some more java libraries were used in our project, the most interesting of them bare the Apache Common Maths. Commons Math is a library of lightweight, self-contained mathematics and statistics components addressing the most common problems not available in the Java programming language or Commons Lang.

Our java servlet was hosted on Openshift. OpenShift is Red Hat’s Platform-as-a-Service (PaaS) that allows developers to quickly develop, host, and scale applications in a cloud environment. We chose this platform as it was very user-friendly, free of charge and integrated into the Eclipse IDE. The URL which needs to be accessed for our API to work is "http://beaconservlet-dpms.rhcloud.com".

All the data exchange between the servlet and both the application and the website is achieved through JSON objects. JSON is an open standard format that uses human-readable text to transmit data objects consisting of attribute-value pairs. As you can see in Figure 5.9, JSON is just a text which contains interesting info.
As you can see, nowadays, smart devices and especially smart phones play a very important role into our everyday lives. So, from the start, we wanted to integrate smart phones into our application and use the iBeacon technology to our advantage. In order to develop our mobile application, we used Xcode IDE which allows developers to write code for iOS devices. In order to use the app on a mobile device, we did not have to purchase an Apple’s developer license, as since the publishing of Xcode 7, every developer can test his app on a mobile device for free.

As you can see in Figure 5.10(a), this is the main page of our application. Here, after you have successfully logged in, the app welcomes you by your name and shows the functionalities this app has. There are four tabs at the down side of the screen. Except the first tab which is the main screen, every other tab contains a page of our application which has a different iBeacon functionality. In our project, we will concentrate only on beacon functionalities.
recommendations and get discounts tabs. There is also the option to use the app without login, but this way, you miss the chance of receiving personalized recommendations. Also, the app provides you the option of registering to Beacommender via the mobile application, and the account you create may be used in the Website as well. The registering form can be seen in Figure 5.10(b). One more thing we have to mention is that a custom message is also presented to the users during their first use of the app. As iBeacon technology needs bluetooth services to be enabled in order for the device to detect beacons, you have to switch on the bluetooth services on your device. In case these services are off, the app prompts you to enable them. In case they are on, the first time you use the app, you will see the message shown in Figure 5.11(a). These message notifies the users that Beacommender app would access your location even when you are not using the app. This function enables the app to receive notifications about nearby deals and discount offers.

![Login Screen](image1.png)
![Application’s Notification](image2.png)

Figure 5.11 iOS Application Tabs - Store sections and products in them

5.4.1 Storing beacon preferences

In our mobile application, we make use of Estimote/iOS SDK. This SDK has two functions for detecting beacons. The first is called "Background Monitoring" and the second "Beacon Ranging". In this section we will explain the first method. You can think of beacon monitoring as a geofence, i.e., a virtual barrier that’s usually defined using a set of geographic coordinates. Moving and out of the area it encloses triggers ‘enter’ and ‘exit’ events, which the app can react to.

In case of iBeacon, the area is defined by the range of one or more beacons. This allows more granularity and precision than regular geofencing; the latter being based on a mix of
Chapter 5: Beacommender System

signals from cell towers, WiFi and GPS. Beacon geofences are also more responsive: ‘enter’ events usually take up to a few seconds to trigger, ‘exit’ events up to 30 seconds. Best part - iOS will keep listening for those beacons at all times—even if your app is not running or was terminated, and even if the iPhone/iPad is locked or rebooted. Once an ‘enter’ or ‘exit’ happens, iOS will launch the app into the background (if needed) and let it execute some code for a few seconds to handle the event. [10].

Figure 5.12  Xcode environment

In Figure 5.12, you can see how the Xcode environment is as well as an example of "Background Monitoring". The code is written inside a class of the project called AppDelegate.swift. This class exists in every project and contains information about the launch of the application. We select this class as AppDelegate’s didFinishLaunching function makes for a perfect spot, since it’ll be called regardless of whether the app is launching into the foreground or background. As you can see, we use a function called beaconManager which takes some parameters in order to be executed. One of the parameters is called region, which is defined by the beacons we want the app to scan. Depending on the region’s major, the app produces a different notification. As you can see in Figure 5.11(b), the same notification message appears on iOS device’s screen when a beacon is detected.

By tapping on this notification, the app transits you to the get discounts tab. We will see later the functionality of this screen. Now let’s get back to Figure 5.12. In the function we mentioned above, the first parameter had a name didEnterRegion. This function is called when the background monitoring first detects a beacon signal and simulates that this is the time you enter the beacon region. So, when the app detects the signal of a distinct beacon, it stores this time to the app’s memory. The duration of the user’s stay there does not affect the app. What it affects is another function, with the didExitRegion identifier. As shown in Figure 5.13, this function has a parameter called elapsed time which is the timeInterval between these two function’s occurrence. When this function is called, we calculate the time interval and along with the beacon’s major and the customer ID (if exists), we make a call to our API(web server) "http://beaconservlet-dpms.rhcloud.com/beacommender/ios". In this request, we send the data in JSON format. Then, our web server receives the data and
5.4 Mobile iOS Application

stores it to our database. Thus, we keep records of ratings as we saw in Figure A.2.

5.4.2 Get Discounts

We reported before that there is a tab in our app called "Get Discounts". In our virtual store, we created two different store sections, called "Apple Corner" and "Samsung Corner" and we assigned a beacon to each of them. In both store sections, there are some product discount deals which may appear to both the website and to our mobile application as well. There are two ways of navigating to the offers menu. The first one is by selecting the appropriate store, whereas the other is by going directly to the appropriate section based on beacon ranging. While monitoring creates a virtual fence to detect when you are moving in and out, ranging actively scans for any nearby beacons and delivers results to you every second. This allows us to find the nearest beacon and transfer you the beacon’s assigned section so as to see the deals. This spatial information is very important, as people who are wandering about looking for deals may be aware each moment about the deals around them. In Figure 5.14(a), we can see the image of different store sections.

The interesting part of our project is that these deals do not appear if the app does not detect the corresponding beacon. This enables the store owners to provide application’s users with unique deals that can only be accessed if they are at the store and use the store’s application. In Figure 5.14(b) we can see the deals available at the Apple Section, whereas in Figure 5.15(a) we can see an image and various other info about the product. The redeem code at the bottom may be used by the user in order to receive the discounted price at the checkout.
5.4.3 Beacon Recommendations

As we said before, the application user will receive recommendations based on his previously stored preferences. These recommendations will use the same user-based and item-based CF algorithms. These algorithms take as input customer_id and beacon_major, along with the corresponding rating, as can be seen in Figure A.2. Let’s suppose the following scenario. There is a store who has three different store sections. The three sections are:

- Apple section, which contains Apple mobile phones, tablets and PC’s
- Samsung section, which contains Samsung mobile phones and tablets
- A general section and it contains general mobile phones, tablets and PC’s.

![Figure 5.14](image)

(a) Different store sections  (b) Products Appearing in Apple Section

A user today has already visited all three sections and has stayed over 3 minutes in each of them, resulting in a rating of 5 for each product in this section. Afterwards, a second user visits the general section and stays over 3 minutes there, thus resulting in a 5 star rating. This creates a correlation between the first user and the second one, as both have rated positively the same section. Then, the second user visits the Apple section. While remaining in the section, he goes to the "Recommend me" tab of the mobile application. As seen in Figure 5.15(b), the second user receives recommendations based on his today’s behaviour. As you can see, the two recommended products are both Samsung products. We can easily understand that, the application, based on user’s behaviour and his stored beacon-preferences, provides the user with Samsung’s section recommendations.
As you can see in Figure 5.15(b), there are two more groups of recommendations in the same page of the application. The other two sections contain the recommendations based on the previous week behaviour of the users while the last one shows the recommendations based on the history of all users. We made this separation as there may be products which are more popular in a special day because of a new arrival or because of a special deal for that day.
Chapter 6

Experimental Evaluation and Results

In this chapter, we proceed with the experimental evaluation of the proposed approach that has been described in the previous chapter. As we discussed, our goal is to compare our proposed method, OmniRank, with other link similarity measures, SimRank and P-Rank. In the first section we provide information about the type, the collection and the preprocessing of data sets. Next, the experimental protocol and setup is presented. Finally, after the completion of the evaluation process, results are being presented and extensively discussed in the last section. Our experiments were performed on a MacBook Pro with 8GB RAM. All algorithms and data set pre-processing were implemented using Java and Eclipse IDE.

6.1 GeoSocialRec dataset

“GeoSocialRec”\(^1\) is an online recommender system for LBSNs where users can get explanations along with the recommendations on friends, locations and activities. GeoSocialRec dataset concerns 149 users who have 595 social ties among them (i.e. friendship network). Also, they have performed 853 check-ins to 438 locations. This dataset is collected between August 2011 and January 2012. In this dataset, we have created a graph where users have edges going to the locations they have been and the locations have edges to the activities they are connected to. This graph has been created based only on positive ratings ($>3$). What is more, we have also added edges among users based on friendship, thus accomplishing to create a unipartite graph. The graph is created using a class called WebGraph, which implements a memory-based Data Structure for storing graphs. This class is created by Bruno Martins\(^2\). We first create the graph based on check-in ratings and then add the friendship links among the users separately.

6.2 HetRec MovieLens dataset

The “HetRec 2011 MovieLens”\(^3\) dataset, which links the movies with their corresponding web pages at IMDb and Rotten Tomatoes movie review system is the second dataset we used; this dataset consists of 855,598 ratings from 2113 users on 10197 movies. In this

\(^1\)http://delab.csd.auth.gr/~symeon/.
\(^2\)http://webla.sourceforge.net/javadocs/pt/tumba/links/WebGraph.html
\(^3\)http://grouplens.org/datasets/hetrec-2011/
dataset, we create a graph again using WebGraph class in java. The first domain of this graph are the users, whereas the second domain is the movies and the third domain is the movie genres. For every movie a user has rated positively, we create an edge from the user node to the movie node. Next, we create an edge from a movie to a movie-genre node, if a movie belongs to that movie-genre. For example, if there is a movie that belongs to Drama, Comedy and Biography genres, then, this movie has an edge from the movie node to all these three movie genre nodes. Finally, the correlation and the edges between the movie genres are based on an external file we have created. From this file, we created a matrix which contains a vector for each movie genre. This vector contains an element for each different movie. This element takes a value of 2 if the movie belongs to the vector’s genre or 1 if it does not. So we create as many vectors as the movie genres and then we use cosine similarity to compute similarities among these vectors (matrix). Then, we calculate the genre-genre similarity. We decided that each genre-genre similarity with a value greater than 0.98 would result in a reciprocal connection between these two genre nodes.

### 6.3 Experimental Setup

In this section, we present the evaluation method that has been followed as well as the performance measures used to evaluate the experimental process. Please note that we test the performance of three different methods (SimRank, P-Rank and OmniRank) combined with the item-based CF algorithm. As we can see in Figure 6.1, we have set different flags into our code in order to easily manipulate the various parameters of our evaluation (for example, the variable NUMOFRECOMMENDATIONS defines the top - N proposals, whereas the boolean geoSocialRecDataset is true when we use the GeosocialRec dataset and false when we use the Hetrec dataset).

![Figure 6.1 Parameters of evaluation in java](image)

#### 6.3.1 Evaluation Method

We have created all our experimental setup into Eclipse IDE for java. We produced all the results in Java console and then we stored this results into an Excel file in order to compare our methods. All dataset entries were inserted into our java classes using java.io package.
In our evaluation method, we have created two different sets based on each dataset we have used. The first one is the training set $E_T$, which is treated as known information. The second one is the probe(test) set $E_P$ which is used for testing. It is clear that no information in the probe set is allowed to be used for the prediction task. It is also obvious that $E = E_T \cup E_P$ and $E = E_T \cap E_P$. Therefore, for a target user we generate recommendations based only on the friends in $E_T$. Each time we separate the entries of the dataset for the evaluation, this separation is random. Thus we can run our evaluation method multiple times creating each time a training set and test set of different values. So, in each run we may get different results for measuring our recommendation techniques performance. In order to evaluate our algorithm, we have to check if a recommendation proposed to a single customer exists in the test set we created. In other words, we check if the customer we are examining has rated positively the recommended item in the test set. We created a separate training set and test set for each user. We also set a threshold for the amount of the rated items by a user. If a user, for example, has rated less than 2 movies or less than 2 locations in the Hetrec and GeosocialRec datasets correspondingly, we leave him out of the evaluation process. In real situations, a user visits an item in order to receive an appropriate recommendation. Likewise, in our evaluation each user receives recommendations for each possible item(movie or location) that is not included in the user’s training set.

### 6.3.2 Performance Measures

The widely-used from the research area of Information Retrieval and Data Mining fields, Precision and Recall metrics are employed as performance metrics for item recommendations. We may assume two classes, one positive and one negative. The positive class represents the prediction of edge existence between a user and an item. Unlike the positive class, we define the negative one that represent the absence of a predicted link between a user and an item. Let’s assume that as true positive are considered the recommended items that have been correctly predicted to be recommended in the top-$k$ list, while we let the true negative to be the items in the list that have been correctly predicted not to be recommended. Moreover, let’s assume that the false positive is the items that have been wrongly predicted to be recommended as user preferences, as well as we let the false negative to be items that have been wrongly considered not to be recommended as preferences but they should have. For a user that belongs to the test set, denoted as test user receiving a list of $N$ recommended items (a top-$N$ list), we define precision and recall as follows:

- **Precision**

  Precision is the ratio of the number of relevant users in the top-$N$ list to $N$. Specifically, those in the top-$N$ that belong in probe set $E_P$ of preferences of the target user.

  \[
  \text{precision} = \frac{|\text{true}_\text{positive}|}{|\text{true}_\text{positive}| + |\text{false}_\text{positive}|} \quad (6.1)
  \]

- **Recall**
Recall is the ratio of the number of relevant users in the top-$N$ list to the total number of relevant users. Specifically, all friends in the probe(test) set $\mathcal{E}^P$ of the target user.

$$\text{recall} = \frac{|\text{true_positive}|}{|\text{true_positive}| + |\text{false_negative}|} \quad (6.2)$$

- **F1 Score**
  
  F1 score (also F-score or F-measure) is a measure of a test’s accuracy. It considers both the precision $p$ and the recall $r$ of the test to compute the score
  
  $$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (6.3)$$

- **Cross-validation**, sometimes called rotation estimation, is a model validation technique for assessing how the results of a statistical analysis can be generalized from an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which it is constructed (training dataset), and a dataset of unknown data (or first seen data) against which the model is tested (testing dataset). The goal of cross validation is to define a dataset to "test" the model in the training phase (i.e., the validation dataset), give an insight on how the model will generalize to an independent dataset

  Precision and recall ratio tend to appear an inversely proportional behavior depicted in Figure 6.2. For instance, in the link prediction problem as we recommend a greater number of items to each user by increasing $N$ in the top-list, we get a higher value of recall, while precision decreases. That is, because in a such greater number of recommendations, we may infer more possible future connections but it is sure that we retrieve an also greater number of irrelevant recommendations.

  ![Typical Precision-Recall curve](image)

  **Figure 6.2** Typical Precision-Recall curve

  Link prediction is highly concerned in particular with the precision metric, without ignoring the recall. There is always a balance point between these two metrics. Best results are expected when the curve approximates the upper right area taking high values in both metrics.
Chapter 6: Experimental Evaluation and Results

6.4 Sensitivity Analysis of the OmniRank Algorithm

In this section, we present the results of the experimental evaluation of the three algorithms. Moreover, we test our newly proposed method (OmniRank) based on different percentages of training set, different fraction of edges used in the auxiliary information and different number of top-\(N\) recommendations.

Before we start, we want to mention that for GeosocialRec dataset, we found that we had to set values 0.4, 0.1, 0.25 and 0.25 for the tuning parameters \(w, x, y\) and \(z\) correspondingly. For the Hetrec Movielens dataset, we found that we achieved the best tuning for our algorithm when we set the values to 0.25, 0.25, 0.25 and 0.25 to the parameters \(w, x, y\) and \(z\) correspondingly.

We start with the presentation of the experiments when we change the Training-set percentages. Figure 6.3 presents 6 diagrams. The first two diagrams (Figure 6.3(a) and (b)) concern the precision we get for different training/test splits. For the GeoSocialRec dataset, as expected, Figure 6.3(a) shows that the best precision is attained as we increase the training set percentage. The same stands for the HetRec MovieLens dataset. In HetRec dataset, we see an even bigger difference in terms of precision between the different training/test splits. This can be explained as the number of entries of the second dataset is greater than the first. Figures 6.3(c) and (d) concern the recall we get when the training/test splits are different. Again, as we expect, the greater the percentage of the training set, the better the recall percentage we get. Finally, we can see at Figures 6.3(e) and (f) how F1-Score differentiates as the training set percentage increases. Again, the best results we get is when we have the highest amount of training set. The difference between the recall percentage for GeosocialRec dataset at Figure 6.3(c) and the HetRec dataset at Figure 6.3(d) can be explained because of the larger amount of movies in the second dataset. As there are more available movies, it is more difficult to predict the correct movie.

Moreover, the same results are presented in Table 6.1. As expected, the higher the percentage of the training set is, the better results we get.

<table>
<thead>
<tr>
<th>Training Set %</th>
<th>OmniRank - Geosocial</th>
<th>OmniRank - Hetrec</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec(%)</td>
<td>Rec(%)</td>
</tr>
<tr>
<td>60</td>
<td>5.25</td>
<td>1.77</td>
</tr>
<tr>
<td>70</td>
<td>6.69</td>
<td>2.69</td>
</tr>
<tr>
<td>80</td>
<td>8.17</td>
<td>3.51</td>
</tr>
</tbody>
</table>

Table 6.1 Results of OmniRank on Geosocial and Hetrec Movielens datasets based on the percentage of the training set

We watch that the precision for GeosocialRec dataset is increased when we take into consideration 80% of the training set. Furthermore, both Recall and F1-score achieve higher scores with the increased percentages of the training set. We get higher precision and recall scores as well as F1-score for the higher percentage of training set. In Table 6.1, you can see the various values for the different percentages whereas in Figures 6.3(b), (d) and (f), the same results appear in a diagram.

The next results we took into account were when we differentiate the amount of information we got in order to calculate the accuracy of location recommendations for GeosocialRec
Figure 6.3 Analysis of Omnirank performance metrics based on different percentages of training set for both datasets. We see the different precision scores (a) and (b), the different recall scores (c) and (d) and the F1-Score (e), (f) for Geosocialrec and Hetrec datasets and the accurate movie recommendations for Hetrec datasets. We assumed that the more information we have about a graph, and more specifically about the links among users in the first dataset and the links among movie genres in the latter, the more accurate our predictions would be. We let the fraction of observed edges (between movie genres and locations) of the training set to vary from 0.4 to 1, as shown in Table 6.2.

In both training sets, as can be shown in Table 6.2, the more amount of information we have about a graph, the more accurate recommendations we are able to provide. As you can see, the performance metrics, for both datasets, increase as the amount of information increases. In Figure 6.4, you can see the difference between precision for the two datasets and the different observed edges we take into account.
Chapter 6: Experimental Evaluation and Results

Figure 6.4 Analysis of Omnirank performance metrics based on different percentages of observed edges between (a) locations for Geosocialrec dataset and (b) movie genres for Hetrec dataset.

Table 6.2 Results of OmniRank on Geocosial and Hetrec Movielens datasets based on the percentage of the fraction of edges observed

<table>
<thead>
<tr>
<th>Fraction of Edges</th>
<th>OmniRank - Hetrec</th>
<th>OmniRank - Geosocial</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec(%)</td>
<td>Rec(%)</td>
</tr>
<tr>
<td>40</td>
<td>3.72</td>
<td>1.55</td>
</tr>
<tr>
<td>60</td>
<td>5.53</td>
<td>2.98</td>
</tr>
<tr>
<td>80</td>
<td>8.07</td>
<td>3.32</td>
</tr>
<tr>
<td>100</td>
<td>8.17</td>
<td>3.51</td>
</tr>
</tbody>
</table>

6.5 Comparison with other methods

In this Section, we will compare OmniRank with P-Rank and SimRank algorithms. The experiments we will conduct should present how well the aforementioned algorithms behave when we increase the number of top-N recommended items. The performance metrics we evaluated were Precision, Recall and F1-Score. We expect to see an increase in the recall and a decrease in the precision as the number of recommendations increases. The curve of Omnirank should look like the one of the diagram showed in Figure 6.2.

In Figures 6.5 (a) and 6.5 (b), there is a visual representation of the Precision/Recall(%) curve for the GeoSocialRec and the HetRec Movielens data sets, respectively. As shown in Figure 6.5 (a), for the GeoSocialRec data set, OmniRank algorithm outperforms SimRank and P-Rank algorithms in terms of precision and recall for all different values (1, 2, 3, 4) of top-N recommended locations. The same results can be observed in Figure 6.5 (b), for the GeoSocialRec data set, for all different values (2, 4, 6, 8) of top-N recommended movies.

More detailed results for the same experiment can be also seen in Tables 6.3 and 6.4, for the GeoSocialRec and the HetRec Movielens data sets, respectively. As shown in both tables, OmniRank achieves better results in terms of precision and recall for all different values of top-N recommended items. In particular, for the GeoSocialRec data set, as shown in Table 6.3, Omnirank achieves a score of 16.67% Precision when we recommend the most similar location to the ones already rated by the target user. This score drops as we
6.5 Comparison with other methods

Simrank
P-Rank
OmniRank

![Graph](image)

**Figure 6.5** Comparing Simrank, P-Rank and OmniRank performance in term of Precision and Recall at top-N recommended items (a) on GeoSocial dataset and (b) on HetRec dataset.

recommend more locations (8.17% at 2, 5.69% at 3 and 5.75% at 4) since it is more possible that we have increased false positives. However, as the number of recommended locations increases, the number of successful recommendations increases as well (recall). Moreover, for the HetRec dataset, as shown in Table 6.4, Omnirank again achieves higher scores both in precision and recall for different values of top-N recommended movies. The precision of Omnirank starts from 10.75% at the top-2 recommendations and drops to 9.13%, 7.75% and 5.96% for the top-4, top-6 and top-8 recommendations correspondingly.

<table>
<thead>
<tr>
<th># Top - N</th>
<th>SimRank Prec(%)</th>
<th>SimRank Rec(%)</th>
<th>P-Rank Prec(%)</th>
<th>P-Rank Rec(%)</th>
<th>OmniRank Prec(%)</th>
<th>OmniRank Rec(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.44</td>
<td>0.99</td>
<td>8.82</td>
<td>1.85</td>
<td>16.67</td>
<td>3.37</td>
</tr>
<tr>
<td>2</td>
<td>6.04</td>
<td>3.47</td>
<td>7.5</td>
<td>3.21</td>
<td>8.17</td>
<td>3.51</td>
</tr>
<tr>
<td>3</td>
<td>5.01</td>
<td>4.43</td>
<td>5.23</td>
<td>4.81</td>
<td>5.69</td>
<td>4.98</td>
</tr>
<tr>
<td>4</td>
<td>5.22</td>
<td>5.41</td>
<td>4.85</td>
<td>5.56</td>
<td>5.75</td>
<td>6.43</td>
</tr>
</tbody>
</table>

**Table 6.3** Algorithms’ comparison on GeoSocialRec data set with different number of Top-N recommendations

Finally, Figures 6.6 and 6.7 present the performance in terms of F1-metric of all three algorithms for both datasets. Please notice that F1-metric combines both precision and recall metrics. Thus, we have a more representative representation of how algorithms perform. As shown again, Omnirank outperforms the other methods in terms of F1-score in both data sets.

In conclusion, we have shown in this Section that OmniRank performs better than the other methods (SimRank and P-Rank). In both datasets, we applied the most common evaluation practises and calculated different performance metrics. The results we got were
Chapter 6: Experimental Evaluation and Results

<table>
<thead>
<tr>
<th># Top - N</th>
<th>SimRank Prec(%)</th>
<th>P-Rank Prec(%)</th>
<th>OmniRank Prec(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>9.52</td>
<td>9.98</td>
<td>10.75</td>
</tr>
<tr>
<td>4</td>
<td>8.18</td>
<td>8.84</td>
<td>9.13</td>
</tr>
<tr>
<td>6</td>
<td>6.89</td>
<td>7.21</td>
<td>7.75</td>
</tr>
<tr>
<td>8</td>
<td>5.01</td>
<td>5.18</td>
<td>5.96</td>
</tr>
</tbody>
</table>

Table 6.4 Algorithms’ comparison on HetRec data sets with different number of Top-N recommendations

Figure 6.6 Algorithms’ comparison on F1-metric vs. different values of top-N Recommended Locations - GeosocialRec dataset

Figure 6.7 Algorithms’ comparison on F1-metric vs. different values of Recommended movies - HetRec dataset

quite encouraging. We found out that the connections among entities of the same domain can play an important role to add value in the information that concerns a graph.
Chapter 7

Conclusion and exploitation of Beacommender

Recommender Systems are very popular nowadays. Almost all e-shops have adopted recommendations at their websites and try to increase their sales based on appropriate proposals of products to their customers. What is more, the huge evolution of smart mobile devices has given everyone the possibility to possess such a device. These devices act more or less as a personal computer allowing users to accomplish everything they can do with a PC. So, every shop aims to have a mobile application as well as an e-shop in order to maximise the customer’s experience. Moreover, every marketing campaign today aims at a more personalised and time-aware content. People want to receive notifications, emails, SMSs and other informational content about special discounts and product deals.

We developed Beacommender System while we had all that in mind. We provide to businesses the option of having a customised website with a unique recommendation engine. Furthermore, we give the firm owners tools, like detailed analytics, in order to understand how much value the recommendations add to their business. Our goal is to combine user preferences of the digital e-shop with the user preferences on the conventional store. By utilising iBeacon technology and with the exploitation of mobile applications, we want to keep track of customer’s preferences in conventional stores. We apply in real life locations (i.e. stores, monuments) an implicit type of rating using beacons. Furthermore, we recommend products to these customers both in right time and right place. Finally, we offer them special discounts for products which are nearby.

Over the Internet, a customer may find different stores and can rate many different products. Our Beacommender project would match well in a shopping centre. First and foremost, as customers of shopping centres are indoor, we maximise the utilisation of iBeacon technology. Furthermore, we can have one application and one website (both with the shopping centre’s name) which would contain many stores from the same shopping centre. Thus, we could recommend various products even from different stores. We could also keep a record of a customer’s preferences over various items like shoes, clothing, sports apparel, electronic devices, etc. The users would benefit from discount prices, would have better user-experience and convenience throughout their purchase process.

Our intention is to develop Beacommender System for both Android and Windows devices as well as iOS. Also, we want to combine ratings and purchases in both e-shops and conventional stores in order to provide recommendations. In real life stores, this can be done by letting the user scan the receipt barcode and provide him with some loyalty
points. Loyalty points is another aspect of Beacommender System which seems interesting. People will download the app and start earning loyalty points. Moreover, by checking in a specific store, the customer will also receive points. Checking in means the customer returns to our store, notifies his friends (if he shares the check in to a social network) and likes the special discounts available only inside the application. This behaviour can be translated to a happy customer, maybe a successful marketing campaign and a better user experience.

Finally, we apply the same recommendations, that can be found in the e-shop in each item’s page in the mobile application. It must become clear to every customer that his purchasing habits in both the conventional store and the e-shop can be combined and that both play an important role in what will be recommended to him. It is crystal clear that most people do not have the time to wander inside shops or search the e-shops infinitely in order to find what they are looking for, so a recommender system may save them time and money.

Moreover, our newly proposed link prediction method, will help us a lot towards this direction. We saw that, in our experimental evaluation, OmniRank outperformed the other state-of-the-art link prediction methods. This can be very useful in recommendation problems and in any other graph problem as well. We showed that in terms of recommendation, there may be link existences between two nodes of the same domain (friendship among customers). These links add valuable info to the edge similarity problem. The great amount of information nowadays makes it difficult to accurately find similar nodes in such a graph and the method we are proposing helps in that direction. This is achieved by calculating the extra information available in our graph. Based on the first dataset, we showed that activities in a certain location performed by users may be more likely among friends. Based on the second one, we found out that the ratings recorded by users to movies may be more similar if the movie genres are similar as well. If we extend this to time-based recommendations, we may find out that a user who happens to buy a certain product in a certain period of time may also buy a similar product in the same time section (in terms of shopping, someone has higher probability to buy two certain items in the same day-excursion to stores). For example, suppose someone who purchases a trousers and combines it with an appropriate belt. In terms of recommendation, this pattern may help our recommender system to propose a belt to whoever seems interested in trousers.

Of course, we restricted our application to tripartite graphs with parallel links. In future work, even larger graphs may be explored in order to find closer similarity relationships. The more data we add to our system, the more accurate our predictions could be, as we showed earlier. Of course, we should always consider the trade-off between time and precision. The biggest problem of our method was the slow execution time, especially in large networks. Although we managed to restrict the calculations only to the corresponding domains and not to all the graph, the execution time was still unacceptable for applying it to a real time recommender system. We show in Section 4.2 that a lot of work is being dedicated in order to reduce the execution time of such algorithms. The first solution to this problem may be the improvement of the calculation time of the algorithm, while the second would be the storing of similarity values in a database. This process would execute offline, once per day. Unfortunately, following this method would result in loss of the capability to process real-time data.
Chapter 7: Conclusion and exploitation of Beacommender

Link Prediction approaches and Data Mining will continue to have a central role in Web since information that is shared online increases each day enormously. Users will need more powerful tools and services in order to manage information in a more efficient way. The exploration and exploitation of multiple sources of information in a unified framework may be the key for more accurate recommendations.
Bibliography


Appendix A

Web Database

As we can see in figure A.1, rs_review table contains:

- a review_id column, which is the unique key in this table,
- a product_id column which is a foreign key to the table rs_product and shows the item that has received a rating,
- a customer_id field which is a foreign key to the table rs_customer and relates this field with each unique customer that has added a rating for a product
- the rating field which depicts the rating value the customer has assigned to a product
- the date_added field which shows the timestamp of the entry.

As shown in figure A.2, rs_ios table contains:

- a record_id column which is the unique key in this table
- a customer_id column which is a foreign key and points to the rs_customer table, showing which customer was at the section and made the entry to the database
Figure A.2  Beacon ratings stored in the database

- a beacon_major column which shows to which store section the customer was, as each beacon’s major is assigned to a section

- a time column, which shows how long the customer has stayed in this section

- a rating column, which goes along with the time. We have concluded that time stayed in each section is assigned to a rating based on the following:

- a time_occured column which shows the timestamp of the event
Appendix B

Web Server

Below, you can see an example of using JDBC to connect to a database

- static final String JDBC_DRIVER = 'com.mysql.jdbc.Driver';
- static final String DB_USERNAME = "u255009881_sterg";
- static final String DB_PASSWORD = "xxxxxxxx";
- static final String DB_HOST = "185.28.20.10";
- static final String DB_PORT = "3306";
- static final String DB_NAME = "u255009881_beaco";

As you can see, we first instantiate the JDBC driver. Next, we define the database’s username, password, host, port and name. Finally, we try to connect to the database. If we succeed, we will be able to run any SQL query we want to the database we connected to.
import java.util.*;
import javax.swing.text.html.InlineView;

/**
 * SimRank is an iterative PageRank-like method for computing similarity.
 * It goes beyond direct cocitation for computing similarity much as PageRank
 * goes beyond direct linking for computing importance.
 * 
 * @author Stergios Chairistanidis
 */

public class simrank {
    static final double dampfactor = 0.80;
    static final double lamda = 0.50;
    static final double alpha = 0.40;
    static final double beta = 0.10;
    static final double gama = 0.25;
    static final double delta = 0.25;

    WebGraph graph;
    WebGraph graphWithParallelLinks;
    Map<Integer, HashMap<Integer, Double>> pRankScores = new HashMap<Integer, Double>();
    Map<Integer, Map<Integer, Double>> parallelInLinks = new HashMap<Integer, Map<Integer>>();
    Map<Integer, Map<Integer, Double>> parallelOutLinks = new HashMap<Integer, Map<Integer>>();
    boolean chechForParallelLinks = false;

    public SimRank(WebGraph graph, WebGraph wgWithParallelLinks, boolean geoSocialRecDataset) {
        this.graph = graph;
        this.graphWithParallelLinks = wgWithParallelLinks;
        chechForParallelLinks = geoSocialRecDataset;
    }

    public void simrankScores(String domain, int domainNodeID, Map<Integer, Double> similarityPerNeighbor, int PICKITEMITEMSIMILARITYMETRIC)
Map<Integer, HashMap<Integer, Double>> similarityScores = new HashMap<Integer, HashMap<Integer, Double>>();
Map<Integer, HashMap<Integer, Double>> oldSimilarityScores = new HashMap<Integer, HashMap<Integer, Double>>();
Map<Integer, Double> neighborFoodSimilarity = new HashMap<Integer, Double>();

int counter = 1;

boolean converge = false;
if (checkForParallelLinks) discoverParallelLinks(graph, graphWithParallelLinks, parallelInLinks, parallelOutLinks);

while (!converge) {
    for (int i = 1; i <= graph.numNodes(); i++) {
        HashMap<Integer, Double> scores = new HashMap<Integer, Double>();
        for (int y = 1; y < i; y++) {
            // We calculate similarity scores only for nodes of the same domain and
            // only between them and the one we are currently viewing
            System.out.println("vrhkame diana "+domain+""+domainNodeID);
            if (graph.IdentifierToURL(y).contains(domain) &&
                graph.IdentifierToURL(i).contains(domain) &&
                ((graph.IdentifierToURL(i).equals(domain+""+domainNodeID)) ||
                (graph.IdentifierToURL(y).equals(domain+""+domainNodeID)))) {
                System.out.println("twra exetazoume tous komvous "+i+" kai "+y);
                if (PICKITEMITEMSIMILARITYMETRIC == 2) {
                    scores.put(y, calculateSimrank(i, y, counter));
                } else if (PICKITEMITEMSIMILARITYMETRIC == 3) {
                    scores.put(y, calculatePrank(i, y, counter));
                } else {
                    scores.put(y, calculateOmnirank(i, y, counter));
                }
            }
        }
        similarityScores.put(i, scores);
    }
}
Appendix C: Evaluation Process Code

```java
for (int key: similarityScores.keySet())
    for (int keyInner: similarityScores.get(key).keySet())
    {
        System.out.println("H omoiota toy komvou "+graph.IdentifierToURL(key)+" me ton komvo "+graph.IdentifierToURL(keyInner) +" einai :");
        similarityScores.get(key).get(keyInner)="|n";
        neighborfoodSimilarity.put(Integer.valueOf(graph.IdentifierToURL(key).substring(domain.length())), similarityScores.get(key).get(keyInner));
        neighborfoodSimilarity.put(Integer.valueOf(graph.IdentifierToURL(keyInner).substring(domain.length())), similarityScores.get(key).get(keyInner));
    }
    if(oldSimilarityScores.equals(similarityScores) || counter == 1)
    {
        converge = true;
    }
    else
    {
        oldSimilarityScores.putAll(similarityScores);
        counter ++;
    }
neighborfoodSimilarity = SortHashMap.sortByComparator(neighborfoodSimilarity);
System.out.println(neighborfoodSimilarity);
System.out.println("0 arithmos twn iterations einai "+counter);
System.out.println("0 arithmos ypopsifiwn geitonwn einai "+neighborfoodSimilarity.size());
similarityPerNeighbor.putAll(neighborfoodSimilarity);
}

public Double calculateSimrank(int link1, int link2, int counter)
{
    if(counter == 0 || link1 == link2)
    {
        System.out.println("Einaste ston ypologismo");
        return calculateSimrankScore(link1, link2);
    }
    else if (graph.inLinks(link1).size() == 0 || graph.inLinks(link2).size() == 0)
        return 0.00;
    else
    {
        counter--;
        double value = dampFactor/(graph.inLinks(link1).size() * graph.inLinks(link2).size());
        double sum = 0.00;
    }
```

for (Object inLinksLink1: graph.inLinks(link1).keySet())
{
    for (Object inLinksLink2: graph.inLinks(link2).keySet())
    {
        int inlink1 = (Integer) inLinksLink1;
        //System.out.println("Komvos "+inlink1);
        int inlink2 = (Integer) inLinksLink2;
        //System.out.println("Komvos "+inlink2);
        sum = sum + calculatePrank(inlink1,inlink2, counter);
    }
    return sum*value;
}

public Double calculatePrank(int link1, int link2, int counter)
{
    double sum = 0.00;
    // first we calculate for the inLinks
    if(counter ==0 || link1 == link2)
        //If we are at the last iteration or if the two nodes are the same
        {
            //System.out.println("Eimaste ston ypologismo");
            sum= calculateSimrankScore(link1, link2);
        }
    //If there is no inLink
    else if (graph.inLinks(link1).size() == 0 || graph.inLinks(link2).size() == 0)
        sum= 0.00;
    else
    {
        //First we reduce the counter
        counter--;
        //Next we calculate the first portion of the equation
        double value = dampFactor/(graph.inLinks(link1).size() *
            graph.inLinks(link2).size());
        //Next we calculate the total similarity
        for (Object inLinksLink1: graph.inLinks(link1).keySet())
        {
            for (Object inLinksLink2: graph.inLinks(link2).keySet())
            {
                int inlink1 = (Integer) inLinksLink1;
                //System.out.println("Komvos "+inlink1);
                int inlink2 = (Integer) inLinksLink2;
            }
        }
    }
Appendix C: Evaluation Process Code

```java
//System.out.println("Komvos "+inlink2);
    sum = sum + calculatePrank(inlink1,inlink2, counter);
}
}
sum= sum*value;
}

//Step 2: We calculate ranking for Outlinks
double sumOutLinks = 0.00;
if(counter ==0 || link1 == link2)
{
    //System.out.println("Estimato ston ypologismo");
    sumOutLinks= calculateSimrankScore(link1, link2);
}
else if (graph.outLinks(link1).size() == 0 || graph.outLinks(link2).size() == 0)
    sumOutLinks= 0.00;
else
{
    counter--;
    double value = dampFactor/(graph.outLinks(link1).size() *
                   graph.outLinks(link2).size());

    for ( Object outLinksLink1: graph.outLinks(link1).keySet())
    {
        for (Object outLinksLink2: graph.outLinks(link2).keySet())
        {
            int outlinkA = (Integer) outLinksLink1;
            //System.out.println("Komvos "+inlink1);
            int outlinkB = (Integer) outLinksLink2;
            //System.out.println("Komvos "+inlink2);
            sumOutLinks = sumOutLinks + calculatePrank(outlinkA,outlinkB, counter);
        }
    }
    sumOutLinks= sumOutLinks*value;
}
return lamda*sum+lamda*sumoutlinks;
}

public Double calculateOmnirank(int link1, int link2, int counter)
{
    double sum = 0.00;

    // first we calculate for the inLinks
    if(counter ==0 || link1 == link2)
```
// If we are at the last iteration or if the two nodes are the same
{
  System.out.println("Einaste ston ypologismo");
  sum = calculateSimrankScore(link1, link2);
}
// If there is no inLink
else if (graph.inLinks(link1).size() == 0 || graph.inLinks(link2).size() == 0)
  sum = 0.00;
else
{
  // First we reduce the counter
  counter--;

  // Next we calculate the first portion of the equation
  double value = dampFactor/(graph.inLinks(link1).size() * 
                 graph.inLinks(link2).size());

  // Next we calculate the total similarity
  for (Object inLinksLink1: graph.inLinks(link1).keySet())
    for (Object inLinksLink2: graph.inLinks(link2).keySet())
      {
        int inlink1 = (Integer) inLinksLink1;
        System.out.println("Komvos "+inlink1);
        int inlink2 = (Integer) inLinksLink2;
        System.out.println("Komvos "+inlink2);
        sum = sum + calculateOmnirank(inlink1, inlink2, counter);
      }
  sum = sum*value;
}

// Step 2: We calculate ranking for Outlinks
double sumOutLinks = 0.00;
if(counter == 0 || link1 == link2)
{
  System.out.println("Einaste ston ypologismo");
  sumOutLinks = calculateSimrankScore(link1, link2);
}
else if (graph.outLinks(link1).size() == 0 || graph.outLinks(link2).size() == 0)
  sumOutLinks = 0.00;
else
{
  counter--;
  double value = dampFactor/(graph.outLinks(link1).size() * 
                        graph.outLinks(link2).size());

Appendix C: Evaluation Process Code

```java
for (Object outLinksLink1: graph.outLinks(link1).keySet())
{
    for (Object outLinksLink2: graph.outLinks(link2).keySet())
    {
        int outlinkA = (Integer) outLinksLink1;
        //System.out.println("Komvos "+inlink1);
        int outlinkB = (Integer) outLinksLink2;
        //System.out.println("Komvos "+inlink2);
        sumOutLinks = sumOutLinks + calculateOmnirank(outlinkA, outlinkB, counter);
    }
}
sumOutLinks= sumOutLinks*value;

//Step 3: We calculate ranking for parallel links
    double sumParallelInLinks = 0.00;
    if(counter ==0 || link1 == link2)
    {
        //System.out.println("Eimaste ston ypologismo");
        sumParallelInLinks= calculateSimrankScore(link1, link2);
    }
    else if (parallelInLinks.get(link1).size() ==0 ||
            parallelInLinks.get(link2).size() ==0)
    {
        sumParallelInLinks = 0.00;
        counter--;
        double value = dampFactor/(parallelInLinks.get(link1).size() *
            parallelInLinks.get(link2).size());
        
        for (Object inParallelLinksLink1: parallelInLinks.get(link1).keySet())
        {
            for (Object inParallelLinksLink2: parallelInLinks.get(link2).keySet())
            {
                int outlinkA = (Integer) inParallelLinksLink1;
                //System.out.println("Komvos "+inlink1);
                int outlinkB = (Integer) inParallelLinksLink2;
                //System.out.println("Komvos "+inlink2);
                sumParallelInLinks = sumParallelInLinks +
                    calculateOmnirank(outlinkA, outlinkB, counter);
            }
        }
        sumParallelInLinks= sumParallelInLinks*value;
    }
```
//Step 4: We calculate ranking for parallel out links
double sumParallelOutLinks = 0.00;
if (counter == 0 || link1 == link2)
{
    //System.out.println("Eimaste ston ypologismo");
    sumParallelOutLinks = calculateSimrankScore(link1, link2);
}
else if (parallelOutLinks.get(link1).size() == 0 ||
    parallelOutLinks.get(link2).size() == 0)
    sumParallelOutLinks = 0.00;
else
{
    counter--;
double value = dampFactor / (parallelOutLinks.get(link1).size() *
    parallelOutLinks.get(link2).size());

    for (Object outParallelLinksLink1:
        parallelOutLinks.get(link1).keySet())
    {
        for (Object outParallelLinksLink2:
            parallelOutLinks.get(link2).keySet())
        {
            int outlinkA = (Integer) outParallelLinksLink1;
            //System.out.println("Komvos "+outlinkA);
            int outlinkB = (Integer) outParallelLinksLink2;
            //System.out.println("Komvos "+outlinkB);
            sumParallelOutLinks = sumParallelOutLinks +
            calculateOmnirank(outlinkA, outlinkB, counter);
        }
    }
    sumParallelOutLinks = sumParallelOutLinks*value;
}

return (alpha*sum) + (beta*sumOutLinks) + (gama*sumParallelInLinks) +
        (delta*sumParallelOutLinks);
//return (alpha*sum) + (beta*(1-alpha)*sumOutLinks) +
//        (gama*(1-alpha)*(1-beta)*sumParallelInLinks) +
//        ((1-alpha)*(1-beta)*(1-gama) * sumParallelOutLinks);
}

private Double calculateSimrankScore(int link1, int link2)
{
    if (link1 == link2)
        return 1.00;
    else
        return 0.00;
Appendix C: Evaluation Process Code

```java
private void discoverParallelLinks (WebGraph wg, WebGraph wgWithParallelLinks, 
    Map<Integer, Map> parallelInLinks2, Map<Integer, Map> parallelOutLinks2 )
{
    for (int i=1; i<=wgWithParallelLinks.numNodes();i++)
    {
        Map<Integer,Double> inParallelLinks = new HashMap<Integer, Double>();
        Map<Integer,Double> outParallelLinks = new HashMap<Integer, Double>();
        for(Object outLinksLink1: wgWithParallelLinks.outLinks(i).keySet())
        {
            if(wgWithParallelLinks.IdentifyerToURL(Integer.valueOf(outLinksLink1.toString())).
                contains(wgWithParallelLinks.IdentifyerToURL(i).substring(0, 1)))
            {
                inParallelLinks.put((Integer) outLinksLink1, 1.00);
                outParallelLinks.put(i, 1.00);
                // wg.addLink(wg.IdentifyerToURL(i),
                //    wg.IdentifyerToURL(Integer.valueOf(outLinksLink1.toString())), 0.00);
            }
        }
        parallelInLinks2.put(i, inParallelLinks);
        parallelOutLinks2.put(i, outParallelLinks);
    }
}
```
Acknowledgements

This thesis would not have been possible without the support of Dr. Panagiotis Symeonidis whose guidance was crucial to its formation and completion. I am utterly convinced that Dr. Symeonidis will continue to conduct primary research on the broader and emerging research area of Data Mining with the same dedication. I hope that we will cooperate again in the future, addressing further challenging issues.

Last but not least, I would like to thank my wife, Christina, for allowing me to realize my own potential and her selfless support and understanding throughout my studies. Finally, I want to express my gratitude to my family, without whom this thesis may not have concluded.