Enhancing a location-based recommendation system by enrichment with semantic web data

Verbesserung der Empfehlungen eines ortsbasierenden Sozialen Netzwerk durch Anreicherung mit semantischen Daten

Diplomarbeit von Max Schmachtenberg aus Darmstadt

Oktober 2012

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Knowledge Engineering

Enhancing a location-based recommendation system by enrichment with semantic web data
Verbesserung der Empfehlungen eines ortsbasierenden Sozialen Netzwerk durch Anreicherung mit semantischen Daten

Vorgelegte Diplomarbeit von Max Schmachtenberg aus Darmstadt

1. Gutachten: Heiko Paulheim
2. Gutachten: Johanne Fürnkranz

Tag der Einreichung:

I would like to thank the Knowledge Engineering Group at Darmstadt University Of Technology for enabling me to write this thesis, especially Heiko Paulheim for his exceptional support and advice. I would also like to thank the projects that enabled me to acquire the necessary data for this thesis, namely LinkedGeoData, OpenStreetMap an Mesowest and the Telecooperation Group at Darmstadt University of Technology for the use of their mining farm. Finally, i would like to thank my family and friends for their support, especially Katrin for always having an open ear for me

Erklärung zur Diplomarbeit

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 31.10.2012

(Max Werner Schmachtenberg)

Abstract

With the rise of Online Social Networks and the availability of location-based services emerged a new type of social network, location-based social networks, which enable users to check-in at a location and share this with their friends and other users. As the number of available locations grows in number, it becomes more and more difficult for users to find those that are interesting to them, making it appealing to use recommender systems to suggest possibly interesting locations to users.
On the other side, the idea of the Semantic Web, emerging in the last decade, makes it possible to infer new knowledge on data and enables intelligent feature mining for enhancing data.
Based on this development, we will try to improve recommendation for location-based social network by semantically enhancing the available data, both with respect to the location of check-in as well as to the act of the check-in itself. The results show that this enhancement leads to better recommendation results, which suggest that incorporating additional knowledge for recommendation seems to be beneficial.

Contents

Nomenclature

app Application

CCDF Complementary Cumulative Distribution Function

FeGeLOD Feature Generation from Linked Open Data

GPS Global Positioning System

HTML Hypertext Markup Language

LGD LinkedGeoData

LOD Linked Open Data

LSN Location-based Social Network

MAE Mean Absolute Error

N3 Notation3

NDPM Distance-based Performance Measure

OSM OpenStreetMap

OSN Online Social Network

POI Point of Interest

RDF Resource Description Framework

RDFS RDF Schema

RMSE Root Mean Squared Error

URI Uniform Resource Identifier

W3 World Wide Web Consortium

XML Extensible Markup Language

List of Figures

List of Tables

1 Introduction

The last decade saw some important technological developments. First, online social networks, which are online services where users can connect, communicate and share information with each other, gained huge popularity and now have billions of registered users. Facebook alone, the largest of them, counts one billion active users per month, nearly 15% of the world population.[1]
Second, a new generation of mobile devices, such as smartphones and tablets, with access to mobile internet saw a wide distribution. These devices have the ability to determine their spatial position in multiple ways, using for example an integrated GPS chip or W-LAN triangularization. This in turn resulted in the possibility to use location-based services on such devices, a kind of service where the data obtained through the mobile network is adjusted to the location of the user.

The combination of these two developments resulted in the emergence of location-based social networks, which later became one aspect of existing online social networks. Their services give the user the ability to declare interesting locations and to "check-in" at them if they are near, sharing this information within the social network and beyond. New location-based social networks like Gowalla[2], Foursquarewww.foursquare.com or Loopt[3] where launched. Later, existing social networks like Facebook and Google+ added similar services to their platform.

For users, one function of these services is to find new and interesting locations in their vicinity. Giving a user personalized recommendations on locations he finds interesting can thus lead to an heightened user satisfaction and an increased use of the service. This is especially important if there is an abundance of available locations to check-in.
In this thesis, we try to enhance recommendation on location for users by adding knowledge to both the locations and the context of a check-in. We hope that by adding knowledge about a location, the system can enhance its understanding of which locations a user likes or does not like while with the aid of context information, the system is able to better understand under what conditions a user might or might not be interested in a location.

When adding data to an existing dataset, the first question is where to acquire the data. For such data, a source can be Linked Open Data, which is data published freely on the internet, using Semantic Web standards. The Semantic Web is a new concept, where data is enriched with well-defined meaning to enhance the human-machine collaboration. The two major aspects of this idea is to formalize data with ontologies, and to create intelligent agents that can consume and use the data from the semantic web.
One possible implementation of such a smart client is to automatically find information about entities and create knowledge about them, which is then usable for machine learning purposes. In the context of location-based social networks, this idea is especially appealing, as data regarding spatial locations are readily available at the Cloud of Linked Data, which is the set of data sources that are published according to Linked Open Data standards. In this cloud, spatial data liked LinkedGeoData[4], constitutes the second largest domain of Open Data. Although suitable context data for check-ins was not found as linked data, we discovered MesoWest, a network of weather stations in North America, which served as basis for an Linked Open Data project. Indeed, weather seems to be relevant context data for check-ins, as the decision of a user where to go is also influenced by the predominant weather conditions.

[1] http://www.guardian.co.uk/technology/2012/oct/04/facebook-hits-billion-users-a-month
[2] acquired by Facebook and shut down
[3] https://www.loopt.com/, now shut down
[4] www.linkedgeodata.org

1.1 Goals

The goal of this this thesis is as following. For a previous acquired dataset from Gowalla, a location-based OSN, we will first identify semantic information for spots and check-ins and examine if they enhance recommendation power.
Our hypotheses are thus as follows:

- (H1) The enrichment of existing location data with semantic data from LinkedGeoData will yield better recommendation results.
- (H2) The enrichment of existing check-in data with semantic data from MesoWest will yield better recommendation results.

For this, we first identify information suitable for integration. For one source, we create a heuristic for record linkage and a heuristic to add a description of a location's environment to its existing data. For context information on check-ins, we plan and perform a crawl for relevant weather data. Based on this, a content-based recommendation system with context-extension is used to test our hypothesis.

1.2 Outline

The rest of this thesis is structured as follows: The next section will give a background in the are of location-based OSN, semantic web and recommender systems, defining basic terminology. Also, work that has been done related to this thesis will be presented. In Section three, the Gowalla dataset will be presented, and an overview over Linked Data will be given, from where we obtained our sources which we use for enhancement of data. Section four will describe how we obtained and linked the different sources to the existing data from the Gowalla dataset, enhancing it in various ways. Our experimental setup for an enhanced recommender will be presented in section six while in section seven, we will discuss the experimental results of our experiments. The final section constitutes or conclusion.

2 Background and Related Work

In this section, we give an overview over this thesis's background, describing and discussing relevant concepts and literature.
Section 2.1 defines location-based online social networks. As the definition is based on the concepts of social networks and location-based services, we will discuss these two concepts first, deriving the definition from these two concepts.
An overview over recommender systems is given in Section 2.2, describing basic approaches for recommending items to users, as well as the idea of context-based recommendation, which allows to consider the circumstances of the recommendation.
In section 2.3, we introduce the idea of the Semantic Web, explain its key standards and how it can help us gain additional knowledge for entities of our existing data.
Lastly, in section 2.4, we give an overview over relevant prior work. This includes research on location-based social networks and research on recommender systems. Special emphasize is put on recommendation for location-based social networks and recommendation in conjunction with semantic technologies.

2.1 Location-based Social Networks

In location-based social networks, people can check-in at locations and share this information with their friends within the network unify aspects from both social networks and location-based services. In the following, the terms "online social network" and "location-based service" are defined. Based on this, a definition for location-based social networks is derived and their key features are described.

2.1.1 Online Social Networks

Online social networks (OSNs) has have attracted billions of users. OSNs like Facebook[5], Qzone[6], Twitter[7] and Google+[8] enable its users to connect and communicate with each other.
Boyd and Ellison define Online Social Networks as follows [19, p. 211]:

> *We define social network sites as web-based services that allow individuals to (1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system.*

According to this definition, a main piece of functionality of an OSN is that users can connect to other users, either bi- or uni-directionally. In the first case, a user sends a friend-request to an other user, which the other may or may not accept. The users who have a bi-directional connection to each other are called, depending on the OSN, for example "friends" (Facebook) or "contacts" (LinkedIn). In the uni-directional case, a connection can be initiated between two users without the need of confirmation from the user with whom the connection has been established.[9] This collection of users is called for example "followers" (Twitter) or "people" (Google+). By connecting to each other, the users form a "social graph", which can be represented by vertices (users) and edges (connections). Depending on whether the connection is bi- or uni-directional, the graph is modeled with undirected or directed edges.

Another main functionality of OSNs lies in the generation of profile pages, which enable users to display various pieces of information about themselves. According to Furth [26, p. 213], profile pages may have the following features:

[5] `https://www.facebook.com`
[6] `http://qzone.qq.com/`
[7] `https://www.twitter.com`
[8] `https://plus.google.com/`
[9] Although, depending on the privacy settings, the consent of the person being followed can be necessary. See for example the privacy settings of Twitter

- **Personal Details**: such as the user's name (or synonym), a profile picture, date and place of birth, gender or relationship status.
- **Contact Details**: the user's mail address, phone number or the residential address.
- **Connectivity Information**: connections that a user has to other users and what type of connection it is (e.g. friend, family, colleagues, best friend, significant other).
- **Personal Interests**: information about the user's personal interests, hobbies, books or bands he likes. Can also include sexual, religious and political views or a list of fan-pages that the user is connected to or groups he is a member of.
- **Curriculum Vitae**: information about the user's educational and employment history, such as attended schools and the current or past field of study plus obtained degrees, current and former employments, including the position, duties and responsibilities. It can also include academic titles, professional certificates, skills and memberships in professional organizations, community and political services.
- **Communication**: the possibility to exchange messages asynchronous via posts on walls and guest book entries. Depending on the privacy settings, this communication can be publicly available or only disclosed to certain other users. Alternatively, asynchronous communication via private messages or synchronous communication via chat may be possible.

Not every OSN has all categories of information available, and OSNs with different focuses may put the emphasis on different aspects. For example in LinkedIn[10], a social network for professional contacts, the curriculum vitae is a central part of the profile, while personal interests may not be of great importance. On the other side, Twitter concentrates on the communication part, while offering a relative sparse profile page for the user to edit and display.
Depending on the OSN, the visibility of the user page can be configured by choosing certain adjustments concerning privacy settings, whose degree of configurability varies from OSN to OSN. Google+ and Facebook for example let the user control which group of persons can see which parts of the profile. Twitter, on the other side, does not allow to differentiate the visibility of the user's profile but does allow to make "tweets" (i.e. messages put on ones wall) seen by everybody or just by followers.[11]

Historically, the first social network sited showed up in the late 90s, with Friendster[12], launched in 2002, being the first of such services that showed a rapid growth. As of 2003, many new OSNs were launched. Apart from more general concepts, specialized services for example targeting business people (LinkedIn, Xing[13]), "passion-centric" sites, that focused on specific interests (such as Dogster for dog owners[14], Couchsurfing for connecting travelers to people with couches for overnight stays[15] and MyChurch for Christian churches and their members[16]) started to grow. Also, websites focusing on media sharing began implementing OSN features. Examples for this are Flickr[17], (photo sharing) Last.FM[18] (music listening habits) and YouTube[19] (video sharing). During that time, OSNs where growing in popularity worldwide, with different country and region-specific OSN (for example LunarStorm[20] in Sweden,

10 https://www.linkedin.com
11 See footnote five
12 http://www.friendster.com/
13 https://www.xing.com/
14 https://www.dogster.com/
15 https://www.couchsurfing.org/
16 https://www.mychurch.org
17 https://secure.flickr.com/
18 https://www.last.fm
19 https://www.youtube.com/
20 Now defunct

Bebo[21] in United Kingdom, New Zealand and Australia) and global OSNs that had different rates of success in various countries (for example Orkut[22] becoming the premier OSN in Brazil and afterwards India). Launched in 2004, Facebook, initially focusing on college students, became the biggest OSN in the world, with one billion active users[23]. Other major OSNs are Qzone (536 million)[24], Twitter (400+ million)[25] and Google+ (400 million)[26].
An important development that has strong influence on OSNs is the increasing use of the Mobile Web. The number of internet-capable mobile devices has been constantly growing over the last years. According to the International Telecommunication Union, by now more than one billion users (around 16% of the world population) have an active mobile broadband subscription, with the the portion as high as 56.5% in developed countries.[27] Thus the mobile use of the internet and thereby also the mobile use of OSNs has a wide distribution and is expected to rise in the future.[28] OSNs reacted with the introduction of ways to access their platforms through mobile devices, either through a mobile version of their web page or through applications for mobile devices. In 2011, one third of all U.S. mobile users accessed social media services through their mobile devices.[29] As of 2012, more than half of all Facebook users access the service through mobile devices, emphasizing the significance of this development.[30]

2.1.2 Location-based Services

Another aspect of location-based social networks is the usage of location-based services. Location-based services, in turn, are based on location services. According to Spiekerman [54, p. 10] they are defined as follows:

> *Location services can be defined as services that integrate a mobile device's location or position with other information so as to provide added value to a user.*

Such services existed since the 1970s, when the U.S. Department Of Defense began operating the Global Positioning System (GPS), a satellite system serving the positioning of people and objects. With the opening of this system to the general public in 1983, many industries began integrating GPS receivers into their devices, using the positioning data to enhance their products and services. Prominent examples for the use of location services are the integration of GPS-based navigational systems into cars, aircraft tracking systems or man-portable GPS devices for navigation or entertainment, such as Geocaching.
By now, the location of devices can be determined in various ways, not only via GPS. Many devices, including smartphones, incorporate a GPS receiver, further adding to the usage of this location service. If the device has a GSM antenna, methods like Cell-ID can be used. Another possibility is to use inhertial navigation, where the movement of the device is used to derive its movement and thus its position. Lastly, local-range technologies, such as Bluetooth, WLAN, infrared or Near Field Communication are available for locating a device's position.[31]
Combined with the introduction of mobile data services, new service concepts that adjust the data obtained through a (mobile) network to the location (i.e. weather information adjusted to the position of the user) became feasible. The distribution of mobile devices and the usage of mobile networks enabled the emergence of location-based Services (LBS). According to Virrantaus et al. [61, p. 66], they are defined as follows:

[21] https://www.bebo.com/
[22] www.orkut.com, operated by Google
[23] http://www.guardian.co.uk/technology/2012/oct/04/facebook-hits-billion-users-a-month
[24] http://asia.cnet.com/blogs/tencents-group-messaging-app-rockets-to-50-million-users-62212469.html
[25] http://www.mediabistro.com/alltwitter/500-million-registered-users_b18842
[26] https://plus.google.com/+VicGundotra/posts/2YWhK1K3FA5
[27] http://www.itu.int/ITU-D/ict/statistics/at_glance/KeyTelecom.html
[28] see for example http://www.ericsson.com/news/1561267
[29] https://mashable.com/2011/10/20/mobile-social-media-stats/
[30] https://www.nytimes.com/2012/05/15/technology/facebook-needs-to-turn-data-trove-into-investor-gold.html?pagewanted=1
[31] https://www.pcworld.com/article/253354/ten_ways_your_smartphone_knows_where_you_are.html

Location-Based services are services accessible with mobile devices through the mobile network and utilizing the ability to make use of the location of the terminals.

LBS can be categorized as person-oriented or device-oriented and push or pull services. [54, p. 14] In person-oriented LBS, the focus lies on the position of the person using a device, for example when a hiker wants to track his position using man-portable navigation system. Device-oriented LBS on the other side focus on the position of the device itself, for example a car or a stolen laptop.
Push services and pull services differ as with push services, a user receives information as a results of his whereabouts without having to actively request it. This can be with (i.e. subscription-based) or without prior consent (e.g. advertising message when entering a new town). With a pull service, the user actively uses an application and "pulls" location-enhanced information from a network, for example when the user wants to find the tram stop that is closest to his current position.

2.1.3 A Definition Of Location-based Social Networks

Based on the previous definitions, a location-based social networks (LSN) can be defined as an online social network that offers location-based services. Current and past popular LSNs are Foursquare[32], Gowalla[33], Google Latitude[34] and Facebook Places[35].
Services offered by a LSN typically include:

- Performing check-ins at points of interest (POIs)
- Notification about check-ins of friends
- Location Games/Achievements
- Recommendations about yet unknown and/or interesting POIs

A central part of LSNs are so called check-ins. When performing a check-in, a user declares that he is currently at a certain POI. In most networks, the user has to be in the vicinity of a POI to be able to check-in, with his position determined through the above mentioned location services. The check-in can happen either manually (the user selects from a lists of places that are in his vicinity and that he is visiting) or automatically (when the user comes within the range of a place the check-in is performed automatically).
POIs are semantic meaningful units which are tied to a certain geographical location. Depending on the network, they are called "places", "venues" or "spots".[36] If a user feels that such a POI is missing in the list of available ones, he can create it himself and provide information about it, such as its name and spatial position, a description and photos. These pieces of information are displayed on profile pages of the POIs along with other information, such as who performed a check-in at the POI.
As POIs are defined by the user and there are no guidelines or quality control, it reflects his opinion and perspective of the POIs interestingness rather that of a neutral perspective. POIs that are primary interesting for the user who created it are for example his home, his working place or his favorite fishing spot. LSNs cope with this differently. Gowalla did not handle such POIs separately, so they were publicly available. Other LSNs enable its users to create "personal" POIs, where performing a check-in is only available to the user and the check-in shared to a more closed circle of friends. Examples for this are private places in Foursquare[37] or the special "Home" place in Google Latitude[38]

[32] https://foursquare.com/
[33] no longer available
[34] https://www.google.de/latitude
[35] https://www.facebook.com/about/location
[36] Facebook:place; Foursquare: venue; Gowalla:spot
[37] http://support.foursquare.com/entries/239884-does-foursquare-have-private-places
[38] https://support.google.com/gmm/bin/answer.py?hl=en&answer=1650346

Similar to OSNs, LSNs give the user the ability to declare a list of friends, usually implemented as bi-directional connections. Users can also create a profile page, showing information about the user and his actions within the network. If a user performs a check-in at a POI, his friends are usually informed about the check-in, although the extend of information shared with friends and other users varies from network to network.
LSNs also offer some sort of location game, which have an achievement character: the user can receive different virtual items or badges by performing activities, such as checking-in at a number of different POIs, performing a check-in at a POI for a certain amount of times or for creating a number of POIs. This items or badges are shown on the user's profile page and reflect the activity of the user, inclining him to perform activities in the network in order to get more badges.
Lastly, LSNs often offer the possibility to recommend unknown and/or interesting POIs to the user. This recommendation is usually based on the position of the user and displays interesting POIs that are in his area or he may ask for POIs in a certain other area. An example for this is the Radar recommendation Engine from Foursquare.[39]

In summary, from the perspective of LBS, a LSN offers person-oriented LBS, as the location of a user lies in the focus of the services. Their services usually are pull or subscription-based push services. Pull services are for example to query for a list of nearby places or the current whereabouts of a friend, whereas push services are for example the receiving of a notification if a friend checked-in at a POI.
From the view of an OSN, LSNs have usually a bi-directional friendship-concept. A major aspect of information sharing between friend is the check-in behavior of friends or, more general, the location of one's friends.
The current and past available LSNs can be divided into two broad categories. First there are networks whose core feature is to offer location-based services for social networking. Such networks include and included Foursquare, Gowalla, BrightKite and Dodgeball among others. Apart from Foursquare, all those services have been acquired by other online social networks over time (Gowalla by Facebook, BrightKite by Limbo, Dodgeball by Google) and shut down (Gowalla in 2012, BrightKite in 2011, Dodgeball in 2009). On the other side, major OSNs by now offer LBS (Facebook: Facebook Places, Google+:Google Latitude, Twitter: Rockdove) in some form. The location-based functionalities are not the central part, but just one aspect of the social network and check-in information are just one kind of information that can be shared with other persons or shown on the profile page.

To what extend check-in information is shared automatically and what control the users has over the sharing of his check-ins depends on the LSN. Li et al. differentiate whether the check-in can be kept private, can be shared only with friends or is shared with all users and if the check-in information can be seen from outside of the network. [40] Gowalla for example did not allow to keep a check-in private, but shared the check-in information with the friends of a user by default. They also allowed to browse who checked-in at a location at what time, regardless of friendship status to the checked-in users. Foursquare as an other example allows the user to keep check-ins private and by default shares a check-in with all friends of a user. Additionally, LSNs allow users to link their updates with external services, so that for example a status update regarding the user's check-in is posted on his profile on Facebook.

2.2 Recommendation Systems

For a user of a LSN, millions of POIs are available for visiting and checking-in. Although he might never get near most of them in his whole life, even if he lives in a town like New York, he still can choose between thousands of POIs surrounding him. This abundance of choices makes it difficult for a user to asses which POIs might be interesting for him and which not. To overcome this problem, a LSN could suggest POIs which are possibly interesting to a user. If such recommendations are personalized i.e.

[39] http://blog.foursquare.com/2011/10/12/the-real-world-now-in-real-time-say-hi-to-foursquare-radar/

the LSN's recommends POIs according to the user's personal interests, the recommendation can become more useful to the user.

For this task, recommender systems can be used. They work based on ratings that users assign to items, expressing their usefulness. For items that a user did not rate, recommender systems estimate a rating, based on his or other user's past ratings of this and other items. If for an item a user did not rate the system estimates a high rating, it can be recommended to him, as he probably would give the item a high rating if he had rated it.
The ratings a user can assign in such systems are usually either scalar, binary or unary:

- Scalar: A numeric or ordinal rating for an item. For example a 1-5 star rating for a POI.
- Binary: for example if he likes or does not like a POI.
- Unary: if he rated the item, for example by visiting or "liking" a POI. Missing means no information available.

Additionally, those ratings might be explicit or implicit. Ratings are explicit if a user expresses the rating through an active action, for example by assigning one to five stars to a POI. With such a rating, one has a direct measurement of the usefulness a user assigns to an item. A disadvantage of explicit rating is that it requires the user to actively cast a rating on the item, which not every user might do in general or only for some items.
Implicit ratings on the other side are derived from the observed behavior of a user, for example if the user has visited a POI or not, or how often he did it. The advantage of this kind of rating lies in the fact that no active action is required by the user, making it more widely available as a rating can be derived from the user's behavior. The downside of an implicit rating is that it is less exact, as the derived rating might not represent the true utility that a user assigns to an item. For example, a user might have visited a restaurant only because he was invited there, not because he enjoys it. As the systems observed that he visited the restaurant, the implicit rating would be a positive for it, which might not reflect the true utility he has for the restaurant.

Defining recommender systems mathematically, let C be the set of all users and S the set of items to be recommended. In our domain, C would be the users of a LSN and the items S the POIs in the database of a LSN. Further, let $u(c,s)$ be an utility function that measures the usefulness of an item s to an user c. Formally, this function assigns a value from a ordered set of ratings R[40] to an user and an item: $u : C \times S \rightarrow R$. When a recommender system is to recommend items to a user, it choses the item $s'_c \in S$ to maximize the user's utility:

$$\forall c \in C, \;\; s'_c = \arg\max_{s \in S} \;\; u(c,s) \tag{1}$$

As a user will only have rated some of the items available, usually a very small fraction, the central issue of a recommender system is that the utility function u is not defined over the whole $C \times S$ space. Thus, for items where a user has not expressed his utility, the rating has to be extrapolated by the recommender system, so that not rated items can be recommended to the user according to Equation 1.

To predict the utility of an item to an user he has not rated, various approaches are available. A widely used classification of recommender algorithms is given by Balabanovic et al. [11]:

- **Content-based recommendations**: where the user's rating for an item is derived from the ratings of similar items the user rated in the past.

[40] i.e. nonnegative integers or real numbers within a certain range

- **Collaborative recommendations**: where the user's rating for an item is derived from other user's rating of this item who had a similar rating behavior like the user in the past.
- **Hybrid approaches**: which combine collaborative and content-based methods.

In some sources, demographic-based recommendation is mentioned as a fourth category. With these systems, recommendation is performed based on features of the user for whom the recommendation is made, for example his gender, age, or relationship status. [20]

In the following, the three main approaches will be described with the help of two examples, adapted from Pazzani et al. [50], where restaurants are recommended to users. For every method, an example use case will be given and the advantages and the limits of the approaches are discussed.

2.2.1 Content-Based Methods

Content-based recommendation systems estimate the utility $u(c,s)$ of an item s for user c based on the utilities $u(c,s_i)$ assigned by the user c to items $s_i \in S$ that are in some ways similar to item s. If similar items are rated high, the systems assigns a high rating to item s and if similar items are rated low, the item gets a low rating assigned.

Consider the example given in Table 1. Here, the rows are words that appear (Y) or do not appear (-) in the description of a restaurant. The last row shows if Jill liked (+) or disliked (-) the restaurant. We want to find out if Jill will like the restaurant "Dolce" or not.
One can see that the word "basil" only occurs in descriptions of both restaurants Jill likes. Further, the words "shrimp" and "salmon" both occur in descriptions of restaurants she likes but also in descriptions of restaurants she does not like. On the other side, the words "noodle" and "exotic" only occur in descriptions of restaurants she does not like.
Taking a look at the words that occur in the description of the "Dolce", the words "basil", "shrimp" and "salmon" occur, which are words that also tend to occur in descriptions of restaurants Jill likes. Further, the words "noodle" and "exotic", which are words that occur in descriptions of restaurants she does not like, are not in the description of the restaurant "Dolce".
Based on this, one can say that the "Dolce" is similar to restaurants Jill likes and dissimilar to restaurants she does not like, as its description contains words that tend to occur in the descriptions of restaurants she likes and it does not contain words that tend to occur in descriptions of restaurants she does not like. Thus one can infer that Jill will probably like the restaurant "Dolce". The example shows an impor-

	Kitima	Marco Polo	Spiga	Thai Touch	Dolce
"noodle"	Y	-	-	Y	-
"shrimp"	Y	Y	-	Y	Y
"basil"	-	Y	Y	-	Y
"exotic"	Y	-	-	Y	-
"salmon"	Y	-	Y	-	Y
Jill	-	+	+	-	?

Table 1: Example for content-based recommendation with spots and their features

tant aspect of content-based recommender systems, namely that such systems need information about the items that are to be recommended. In the example above, this information was the descriptions of restaurants. Other information for restaurants could be its spatial location, its size, the opening hours.
The available information are then passed through a transformation function $Content(s)$ that creates a feature vector from these information, representing the item.
Depending on if the information available is either structured, semi-structured or unstructured, it may

need to be pre-processed in order to transform it into a feature vector. For structured information, usually no transformation is required. Semi- or unstructured data on the other side needs to be pre-processed. For text data, this incorporates some natural language processing. For example free-text descriptions of an item can be broken down into n-grams, i.e. overlapping sequences of n words. Also, words can be stemmed, where words are reduced to their root form to reflect common meaning. When stemming words like "computation" "computer" or "computers", they are reduced to "compute", their root form. Also, stop word lists may be used, which filter out meaningless words such as "a", "the" or similar. [51] In the example in Table 1, we simply used words that appear in the restaurant's description to characterize it, i.e. an uni-gram tokenizer and a stop word list, filtering out uninteresting words.
With the items represented as feature vectors, for every user a classifier is trained to generate a content-based profile. This is done by presenting an item's feature vector and its rating assigned by the user as class to a classifier. For Jill, we would train the classifier by showing him feature vectors for "Kitima" and "Thai Touch" with a negative rating and the vector from "Marco Polo" and "Spiga" with a positive rating. An often used classifier is Naive Bayes, but also others, like linear classifiers or rule learners may be used. [51]. Such a classifier creates a content-based profile $ContentBasedProfile(c)$, which can be defined as a vector of weights $w_{c1}, \ldots, w_{ck}$ where each weight w_{ck} denotes the importance of the feature vector's features. For example Jill's profile would have a negative weight for the word "noodle", as she rated restaurants with this word in their descriptions negative, and a positive weight for the word "basil", as she assigned a positive rating to restaurants whose description incorporates this word.
If now an unrated item s should be rated for a user c, its feature vector is presented to the classifier which estimates the rating for the item.

$$u(c,s) = rate(ContentBasedProfile(c), Content(s)) \tag{2}$$

On this basis, it can be decided if it may or may not be recommended to the user.

Limits of Content-Based Recommender Systems

As content-based recommender systems operate on user's preferences for items and their feature vectors, information about those items needs to be available. In some cases, it might be difficult to either obtain these information or to transform them into feature vectors. For example, some information available about a restaurant, like the available food or the opening hours might be easily acquired, while information regarding the atmosphere, the quality of service or how "nice" are much harder to find. Although descriptions by users in natural language may be used, it poses an effort to process them in a way as to create meaningful features. Then, they still might not give an formal indication of the service's quality as they just express how the user described a place.
Describing items as feature vectors also means that two items are the same if they have the same feature vector and that the classification of an item is only based on available features. If for example there are two restaurant with exactly the same feature vector, but one is generally favored by the users because of its exceptionally good service while the other one disliked for its poor service, the recommender system is not able to differentiate them, as they lack features describing their difference. Another example is if a user favors coffee shops with strictly organic coffee. If there is no feature describing if the coffee is organic or not, the recommender system will not be able to recommend him coffee shops with organic coffee.
Another issue with content-based recommender systems is that they have tendency for overspecialization, meaning that the system can only recommend items that are similar to the users's preferred items. A user will not get recommendations for items whose features are not similar to the ones he rated, regardless if he never rated an item with the feature or if he showed a distaste for items with this feature. For example a user might never get a recommendation for a Russian restaurant because he never showed any preference for Russian cuisine. This might be either because he does not like Russian cuisine or because he never tried it.
Finally, content-based recommender suffer from a cold-start problem. This means that a user needs to

have given enough ratings in order to enable the recommender to create a meaningful profile of his preferences.

2.2.2 Collaborative Methods

Collaborative filtering on the other side predict the utility $u(c,s)$ a user c would assign to an item s not rated by him based on how other users $c_j \in C$ that are similar to user c rated item s. To asses the similarity of two users, the system evaluates how similar their ratings on other items than s are. In our domain, this means that a user gets recommendations for POIs that are preferred by users who visit share the user's preferences and aversions on other POIs.

Consider the example given in Table 2. Each row represents a user, each column represents a restaurant. The symbol at the intersection of a row and a column show if user c liked ("+") or did not like ("-") restaurant s after visiting it. Jill has not visited the restaurant "Dolce" yet, and we want to see if it should be recommended to her or not (i.e. if she will probably like it or not).
One can see that Karen and Jill have a similar taste regarding restaurants, as they liked or disliked the same restaurants. Chris also seems to have a taste similar to Jill's as except for the "Thai Touch", his ratings are the same as Jill's. Those two users rated in favor of the "Dolce". Opposing to this, Lynn and Mike have a low similarity to Jill. Each of them only voted for one restaurant the way Jill did, in the other three cases they disagreed with her. Their votes on the "Dolce" are both negative.
Because Karen and Chris, whose tastes are similar to Jill's, tend to like the "Dolce" and Lynn and Mike, who have a dissimilar tastes compared to Jill, tend to dislike the "Dolce", one can infer that Jill will like the "Dolce" and thus it can be recommended to her.

Collaborative Filtering approaches can be divided between memory-based (or heuristic-based) and

	Kitima	Marco Polo	Spiga	Thai Touch	Dolce
Karen	-	+	+	-	+
Lynn	+	+	-	+	-
Chris	+	+	+	-	+
Mike	+	+	-	+	-
Jill	-	+	+	-	?

Table 2: Example for collaborative filtering with matrix of users and ratings for restaurants

model-based approaches.
In the example above, we used a memory-based approach. The value of an unknown rating $u(c,s)$ is derived by taking from all users that casted a vote on s those N users that have the most similar rating behavior to user c. Their ratings for item s is then aggregated to a predicted rating for the item and user c. To compute the similarity between users, correlation-based approaches like Pearson's Correlation Coefficient or cosine-based similarity are widely used. [2]
When the N most similar users are found, their ratings are combined. Various aggregate functions have been developed for this, from simply taking the average of all ratings to more sophisticated ones, for example where the rating of more similar users have more importance. One can refer to Schafer et al. for an overview of such methods.[58, p. 302f.]
A variant of this approach is the so called item-based approach. Here, instead of recommending an item based on the rating of other users similar to user c, items are recommended based on the rating a user gave to other items that are similar to the item s. Contrary to content-based approaches, similarity between items is assessed on the rating structure of the items. This means that a user gets an item s recommended if he voted in favor of other items that other users rated similar like item s. One can think of this as a transposition of the approach described above, where items and users are interchanged.[58, p. 304]

A disadvantage of memory-based approaches is that for estimating a user's rating for an item, first the most similar users who voted on *s* have to be found. This can be especially costly if the item is often rated, as a lot of users are possible candidates for being one of the most similar users to user *c* and also makes it necessary to have user profiles in-memory to allow fast rating predictions.
A way to overcome this problem is to use model-based approaches for collaborative filtering. With this approach, the collection of ratings is used to learn a model, which is used for rating prediction. For generating the model, various machine learning approaches can be used, as described in [2]. For example, a probabilistic model could be learned on the rating of users for items which is then used to classify an item, given a user and its ratings on items. Or using a rule-learner, rules similar to "if the user rated items *x* and *y* high, he will rate item *s* also high" can be learned. This approach has the advantage that the user-rating-matrix does not need to be in memory, as only the model needs to be available. A downside is that the models are expensive to create, why they are only updated after a certain amount of time. Also, there seems to be a decrease in predictive accuracy.[41]

Limits of Collaborative Filtering

Collaborative filtering methods have been widely used as recommender systems. Their key advantage is that they do not require knowledge about the items,as they only rely on the user-item rating matrix, enabling them to deal with any kind of content. As they rely on the rating of other users, they are also able to capture features of items that are hard to analyze in a content-based setting, for example "quality", which a collaborative filtering approach captures through the opinion of other users.
A problem of collaborative recommender systems is the so called cold start problem. In order to recommend items to a user, he first needs to casts some ratings, so the system is able to find other users which are similar to him. If an item *s* should be rated, but the similarity between user *c* and all other users is low due to his lack of ratings, the recommendation quality is low.
A similar problem also exists for new items. If the rating of an item should be estimated for a user, first a substantial number of other users need to have rated it already, as otherwise not enough users are available that are similar to the user for which a rating should be derived. In the most extreme case, items that where never rated will not be recommended at all, as there are no similar users to user *c* that serve as the basis for calculating the derived rating.
Related to this is the issue of sparsity of the user-item rating matrix. The number of ratings in a collaborative system is usually small compared to all possible ratings. Users will have rated only a very small fraction of all items, making the matrix sparse i.e. with lots of empty entries. The higher the sparseness, the harder it becomes to find similar users to a given user and item, which in turn influences the ability to predict the ratings accurately.

2.2.3 Hybrid Approaches

A way to overcome the shortcomings of both content-based and collaborative recommender systems is to use hybrid methods, which combine aspects from both approaches. Hybrid approaches can be classified as follows [2]:

1. Implementing collaborative and content-based methods separately and combining the predictions.
2. Incorporate some content-based characteristics into a collaborative-based approach.
3. Incorporating some collaborative-based characteristics into a content-based approach.
4. Constructing an unifying model that incorporates both content-based and collaborative characteristics.

With approach one, the results of one or more collaborative and content-based recommender systems are combined. The combination of ratings can be done in different ways, for example by taking the

41

average rating of multiple recommendations[20, p. 381], or by using a voting scheme. [50, p. 404f.] This way, recommendations based both on the content and other user's opinion can be incorporated.
With approach two, a collaborative recommender system is used while the content-based profiles of each user are maintained. For example, the similarity of users can be computed based on their content-based profiles instead of their ratings. [50, p. 401f.] Another approach is to add ratings for items by users based on a content-based prediction. [52] Such approaches often aim at avoiding sparsity-related problems, as in the first case the sparsity of a matrix is of no issue while in the second case, the sparsity is lowered.
The third approach is based on adding collaborative features to the feature vector of an item, for example by adding a feature describing the average ratings of an item or features describing if the N users most similar to a user like or do not like the item. This approach aims at adding features to an item's description that represent other user's opinion, which can be a meaningful feature.
Finally, some methods have been developed to unify collaborative and content-based methods, making up the last class. Examples Ansari et al. combine user-profiles, item-profiles and the interaction between each others into a unified probabilistic model. [7]

2.2.4 Context-Based Methods

Traditional recommendation systems as described above only deal with two types of entities, users and items, why they can be called two dimensional (2D) recommender. Their model suggest that, regardless of the context a rating is casted, the user will always assign the same utility to an item. But for example when recommending POIs to a user, he might assign no utility to a restaurant if he as already eaten, but a high utility if he is hungry. Or when recommending a TV show to watch, a user might be more interested to watch news in the morning and movies in the evening, giving both classes of TV shows different values depending on the time of day. Such examples show that a the utility of an item can be depending on the context and thus recommendation should adapt to this. If the context is considered, the predictive power of a recommender systems can be further enhanced.
For the abstract concept of context, many different definitions exist, depending on the domain.[4, p. 219] As we deal with LSN data where users check-in into spots, one can think of context as the circumstances under which a check-in is performed. Such contextual information can refer to the check-in itself, such as where it happened, the time when it happened or the weather when the check-in occurred, but it can also include factors describing the condition of the user, such as who was with or near him, or his condition like his stress level during the check-in. Also, factors referring to the checked-in spot can be included, for example how long the spot has already opened or how many persons checked-in on that day.
Thus, considering context, the task of recommending i.e. given a rating for a user and an item can be expanded to derive the utility for an user, an item and a context: $Users \times Items \times Context \rightarrow Ratings$.

Adomavicius et al. define three different paradigms for incorporating context [4]: contextual pre-filtering, contextual post-filtering and contextual modeling.
Contextual pre-filtering uses contextual information to pre-select or construct the most relevant ratings or items to be used in a traditional recommendation system. If ratings are pre-filtered, only ratings associated with a certain context are used for training a traditional 2D recommender. If for example a person want to get a recommendation on a movie for the weekend, only ratings given on a weekend may be considered for recommendation. This also means that if the recommendation is model-based, different models for different contexts are generated, for a example a model for movie recommendation on weekend days and one for week days.
If the pre-filtering is item-based, given a context, only certain items are considered for recommendation. If for example a user is standing on the Empire State Building in New York and wants to go dining, only restaurants in his vicinity are of value for him as they are near, although when considering all restaurants in the US, the recommender might find ones that better suit suit the user's preferences. This method helps reducing the workload of recommendation, as less items are considered, in our example

only those near the user.
Contextual post-filtering on the other side begins with performing a recommendation with the classical 2D-approach. After generating a list of recommendations, they can be adapted to the context by either filtering out inappropriate items or by re-weighting the score of items with their appropriateness, given the observed context. If for example there exists a system that recommends activities to a user, a list of recommendations can be generated, which are then re-weighted based on the current weather, for example down-ranking or even removing outdoor activities if it is raining or up-ranking a visit to the beach if it is sunny.
The third paradigm is called contextual modeling. It uses contextual information directly in the recommendation function and is, opposed to the other two approaches, a true multidimensional recommendation function. A possibility is to extend neighborhood-based approaches discussed in Section 2.2.2 for recommendation in a multidimensional vector space. The rating is then a point with "coordinates" user, item and context and based on this, similarities between users can be assessed and ratings derived.
Alternatively, model-based recommendation techniques like the unifying hybrid model proposed by Ansari et al.([7]) can be extended for multidimensional recommendation. See Adomavicius et al. ([4]) for details.

2.3 Semantic Web

When trying to find additional information about entities of a LSN, a possibility is to use data from the Semantic Web. Tim Berners-Lee, the founder of the World Wide Web Consortium (W3C) , who coined the term "Semantic Web" and defined it as follows [16]:

> *The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*
>
> *...a web of data that can be processed directly and indirectly by machines.*

The idea of the Semantic Web arises from the fact that the current web is tailored for human consumption in mind. Web pages are usually formatted via Hypertext Markup Language (HTML), which only describes how the content should be displayed, but not its meaning. Although tools like search engines exist to help user find information, they still suffer from principal problems that could not be completely solved, only mitigated. [8, p. 1] Those problems are:

- High recall, low precision: Even if the most relevant pages are retrieved, lots of less and non-relevant pages are also retrieved. This effect can me mitigated by ranking the results, but still the principal problem prevails.
- Low or no recall: It may also happen that no pages can be retrieved at all. Although less of a problem with todays search engines, this is still an issue.
- Sensitivity of vocabulary: As search engines analyze the words that are in a document, if a terminology is entered that cannot be found on a web page, it will not be retrieved.
- Returns of single web pages: As search engines return a list of web pages, information that is spread over multiple web pages has to be retrieved from all pages manually and put together.

Although search engines may help, the main burden of searching and combining information still lies on the user.
Consider the typical task of finding a restaurant based on some criteria. For example, you want to go dining in a town where you newly moved to. Preferably, the restaurant should be near to your current location and have less than twenty tables, as you don't like it crowded. Additionally, you might want to order a vegetarian meal.
With the current state of the web, for finding a suitable restaurant, you would first have to search for

restaurants that are in the vicinity of your location. Today, there are some services that help you with this task, for example Yelp[42] or Google Maps[43], which show you all POIs at a certain location whose description match a certain keyword, for example "restaurant". Next, you would have to find restaurants with less than twenty tables that serve vegetarian cuisine. This can easily be a time-consuming task, as you have to visit the websites of the restaurants, look at their menu and assert if at least some dishes are vegetarian. Also, you would have to search for information about the number of tables. The whole process of finding and combining information will take some time.

To prevent such situations and ease the task of information retrieval, it was proposed to make the Web more machine-processable by extending it with the Semantic Web. It is envisioned that this semantic web can be used by intelligent software agents, using the machine-processable information to ease the task of information retrieval and inferring new knowledge.
In our restaurant example, with a established Semantic Web and an intelligent software agent, one could define a query to give a list of restaurants with less than 20 tables within a certain area that have vegetarian dishes on their menu. Based on this, the agent would search the Semantic Web, assembling and organizing the information autonomously. This would save a huge amount of time to the user and would also be more thorough than manual information retrieval. To make this idea come true, the W3C started the Semantic Web Initiative, which is to set standards for the domain of the Semantic Web. In the following, key concepts of the Semantic Web will be described with the help of the example outlined above.

2.3.1 Key Concepts of the Semantic Web

The basic concept of the Semantic Web is a resource. A resource can basically be anything, for example a person, a place or an object, but also something abstract like a theory or a receipt. [8, p. 63] In our example, resources we will talk about are mainly restaurants and dishes.

If we talk about a resource, for example the restaurant Shah in Sunnyvale, we could simply refer to it by its name, "Shah", assuming our conversational partner knows which restaurant we talk about. But as there are probably many other restaurants with the same name in the whole world, this might be a fallacy. To a human, we would usually give additional information to uniquely identify "our" restaurant, for example by saying "the one in Sunnyvale". Another way to uniquely refer to "our" restaurant, which is more compatible for use with computers is to assign an Uniform Resource Identifier (URI) to it. For example, the restaurant "Shah" we talk about might have the URI "`http://www.places.com/establishment/shah`" and another restaurant named "Shah" might have the URI "`http://www.places.com/establishment/shah_bangalore`". If we later express knowledge about "our" restaurant, we simply can refer to it via its URI, avoiding ambiguity.

In the Semantic Web, information regarding a resource is expressed using the Resource Description Framework (RDF).[44] RDF behaves to the Semantic Web like HTML to the Web. As HTML describes how content is formatted for displaying it to the user, RDF describes a resource, tailored for machine consumption. RDF is by standard serialized in XML, but other formats with readability for human readers in mind have been developed, for example Notation3 (N3).[45]
In RDF, knowledge about a resource is expressed via a set of RDF statements. [66, p. 25] A single RDF statement is of format:

resource (subject) + property (predicate) + value (object)

[42] `http://www.yelp.de/`
[43] `https://maps.google.de/`
[44] W3C Reommendation on RDF model ans syntax specification: `http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/`
[45] Team Note from W3C: `http://www.w3.org/TeamSubmission/n3/` Note that this is no recommendation.

The subject is the resource we want express knowledge about. In our example, it is the restaurant "Shah" with the URI "`http://www.places.com/establishment/shah`".
The predicate is also resource that can be used as a property to describe some specific aspect, characteristic, attribute or relation of a given resource. For example, the resource `http://www.places.com/ontology#numberoftables` describes the number of tables that a restaurant has.
Finally, the object is the value that is assigned to the subject via the predicate. It can be either a literal (e.g. a string, number or date) or another resource.
In our example, if we wanted to express the knowledge that Shah has ten tables, we could express it through the triple described in Table 3. With a set of such triples, encoded in XML, we can now express

Subject	`http://www.places.com/establishment/shah`
Predicate	`http://www.places.com/ontology#numberoftables`
Object	10

Table 3: Example of a triple describing the restaurant "Shah"

our knowledge about the restaurant. Apart from the number of tables, we also know the dishes that are served at Shah's, which are "Achar Gosht", "Chana Masala", "Maghaz Masala" and "Kheer".

```
1 <?xml version="1.0"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3          xmlns:est="http://establishments.com/ontology#">
4 <rdf:Description "http://www.places.com/establishment/shah">
5   <rdf:type rdf:resource="est:Restaurant"/>
6   <est:numberOfTables>10</est:numberOfTables>
7     <est:served_dishes>
8       <rdf:li>http://foods.com/achar_gosht</rdf:li>
9       <rdf:li>http://foods.com/chana_masala</rdf:li>
10      <rdf:li>http://foods.com/maghaz_masala</rdf:li>
11      <rdf:li>http://foods.com/kheer</rdf:li>
12    </est:served_dishes>
13  </rdf:Description>
14 </rdf:RDF>
```

Listing 1: RDF/XML data for restaurant Shah

Listing 1 shows the content of an RDF file for the restaurant Shah. Line two and three is an opening tag for RDF data and defines xml-namespaces for the RDF-vocabulary and for the ontology `http://establishments.com/ontology#` (more on that later). Line four expresses the fact that the resource we talk about in line 4 to 13 will be our restaurant "Shah". In line five it is stated that "Shah" is of type restaurant[46]. Line six states that the restaurant has 10 tables. Finally in line seven, a list is started, enumerating the dishes the restaurant is serving. Lines 8 to 11 are the URIs of the dishes that are served. Note that we both have properties that have a literal as object (numberOfTables in line six) as well as properties whose objects are another resource (the served dishes in lines 8 to 11).

Now we take a look to see what information may be found about the dishes. By navigating to the URIs of one of the dishes, for example `http://foods.com/chana_masala`, we can get another RDF document, describing the dish. The following listing is an excerpt of this file:

```
1 <?xml version="1.0"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3          xmlns:foods="http://www.foods.com/ontology#">
4 ...
5   <rdf:Description "http://foods.com/chana_masala">
```

[46] To be more precise of type `http://establishments.com/ontology#Restaurant`

```
6   <rdf:type rdf:resource="foods:dish"/>
7     <foods:ingredient>http://www.ingredients.com/garbanzo_beans</foods:ingredient>
8     <foods:ingredient>http://www.ingredients.com/onion</foods:ingredient>
9     ...
10    <foods:vegetarian>yes</foods:vegetarian>
11    ...
12  </rdf:Description>
13 </rdf:RDF>
```

Listing 2: RDF/XML data for dish Chana Masala

One can see in Listing 2 that two of the ingredients of Chana Masala are garbanzo beans and onion (the URIs of these ingredients at line seven and eight). Line 10 states that the dish is vegetarian, the information we searched for.
With this RDF files, describing a restaurant with links to RDF files describing the dishes served, we could already use an intelligent agent to list restaurants for us that serve vegetarian meals. If we queried this agent for restaurants with less than twenty tables and vegetarian dishes, it would get a list of restaurants (maybe from a semantic version of Google Maps), traverse the list, looking for restaurants with less than twenty tables. For every restaurant that meets this requirement, it would traverse the list of dishes and see if there is a dish for which the value of the property "vegetarian" is "yes". In a more advanced version, such an agent could even traverse the list of all ingredients for every dish, checking if the ingredient is of type meat or fish, and conclude itself if the dish has no meat and is thus vegetarian. For this, it would have to know what a vegetarian dish is (i.e. a dish without ingredients of type meat or fish). As the concept of "vegetarian meal" can also be expressed in RDF, the agent could process the concept, apply it on the dishes in question and would thus infer new knowledge from the existing data by finding out itself if a dish is vegetarian or not. All this would happen without the need of work being done by the user, except formulating the query.

So with RDF, one can assign metadata to a resource which can be processed by a computer. But what would happen if one would add the following statements to the description of the restaurant:

"`http://www.places.com/establishment/shah`" has `vehicle:HorsePower` value of "50"
"`http://www.places.com/establishment/shah`" has `est:NumberOfTables` value of "potato"

Those statements are clearly absurd. First, a restaurant has no horsepower and second, a restaurant cannot have the string "potato" as number of tables. But, unlike us, a computer does not know that and dutifully processes that Shah has 50 horsepower and "potato" number of tables.
These examples show that some kind of rules are necessary to define which properties are allowed for a resource and what kind of values are allowed for the properties. For a restaurant the number of places, the dishes and the address seems valid, horsepower obviously not. Also, the values allowed for the property "numberOfTables" should be a number greater or equal zero, not a string. Such rules can be expressed using ontologies. Ontology vocabularies or short ontologies are an "explicit and formal specification of a conceptualization"[14]. Here, we explicitly formalize our conceptualization of a restaurant. An ontology not just describes what properties an item can have, but it also defines a taxonomy of items. A taxonomy defines groups of items and their relation to each other. We have seen that Shah is a restaurant. More generally, a restaurant is a kind of establishment. Establishments usually have for example an owner and an address. An other example for establishments could be a book store.
As a restaurant is a subtype of an establishment, it "inherits" the properties from the class. So a restaurant both has a number of tables and a list of dishes, but also an owner and an address. The same is true for a book store. A book store is not a restaurant, so it cannot have dishes, but it can have a list of books. As it is also an establishment, it also has an owner and an address.
So we know restaurants and book stores are types of establishments. A restaurant usually has a list of dishes and a number of tables. A book store, in turn, has books, something a restaurant usually not has. An establishment has an owner and an address.

Taken together, this already forms a small ontology, which is shown in Figure 1.

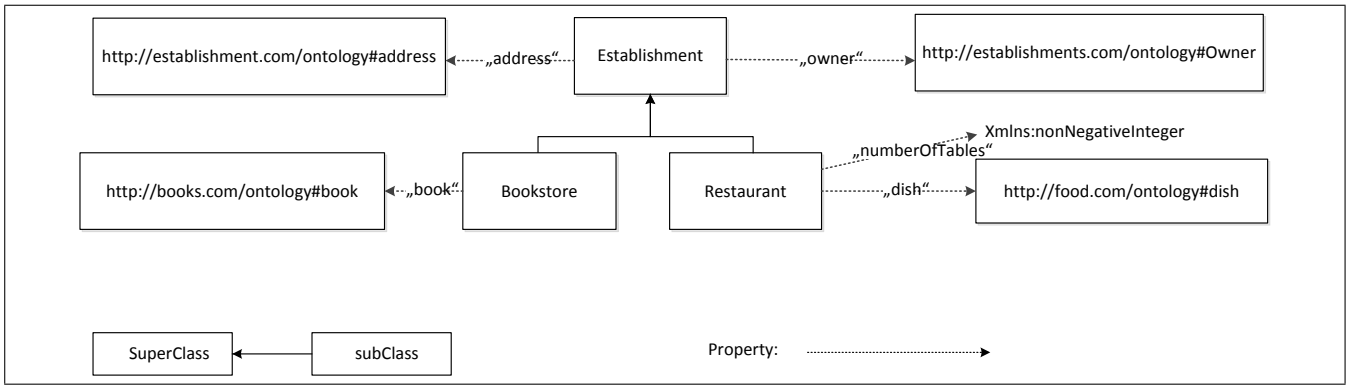


Figure 1: A simple example ontology

Such an ontology can be written down (serialized) in RDF Schema, abbreviated RDFS. RDFS is based on RDF, but defines the vocabulary to express knowledge about classes, subclasses and properties.[47] In Listing 3, the ontology described above is serialized in RDFS.
An alternative to RDFS is to use Web Ontology Language(OWL) or OWL2 respectively.[48] OWL has RDFS as a subset, but is more expressive than the rather limited RDFS. With OWL, one can for example express cardinalities for objects, such as defining that only one RDF statement with the property “numberOfTables” is allowed for a restaurant. Here, for simplicity, we restrict ourselves to RDFS.

```
<?xml version="1.0"?>
<rdf:RDF xlmns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://places.com/place.rdfs"/>

         <rdfs:Class rdf:ID="Establishment">
         </rdfs:Class>
         <rdfs:Class rdf:ID="Restaurant">
           <rdfs:subClassOf rdf:resource="#establishment"/>
         </rdfs:Class>
         <rdfs:Class rdf:ID="Bookstore">
           <rdfs:subClassOf rdf:resource="#establishment"/>
         </rdfs:Class>

         <rdf:Property rdf:ID="owner">
           <rdfs:domain rdf:resource="#establishment"/>
           <rdfs:range rdf:resource="#owner"/>
         </rdf:Property>
         <rdf:Property rdf:ID="address">
           <rdfs:domain rdf:resource="#establishment"/>
           <rdfs:range rdf:resource="#address"/>
         </rdf:Property>
         <rdf:Property rdf:ID="numberOfTables">
           <rdfs:domain rdf:resource="#Restaurant"/>
           <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
               nonNegativeInteger"/>
         </rdf:Property>
         <rdf:Property rdf:ID="dish">
```

47 W3C Recommendation: `http://www.w3.org/TR/rdf-schema/`
48 W3C Recommendation for OWL 1: `http://www.w3.org/TR/owl-ref/` W3C Recommendation for OWL 2: `http://www.w3.org/TR/owl2-overview/`

```
28            <rdfs:domain rdf:resource="#Restaurant"/>
29            <rdfs:range rdf:resource="http://www.food.com/ontology#dish"/>
30          </rdf:Property>
31          <rdf:Property rdf:ID="book">
32            <rdfs:domain rdf:resource="#Bookstore"/>
33            <rdfs:range rdf:resource="http://books.com/ontology#book"/>
34          </rdf:Property>
35
36  </rdf:RDF>
```

Listing 3: RDFS serialization of the example ontology

The listing above shows the simple ontology that we developed, expressed in RDFS. Line three defines a xml-namespace for RDFS. In line 6 to 13 we define the classes described above. for example in line 8 to 10, we define the class "restaurant", expressing that "restaurant" is a subclass of "establishment" in line nine. In lines 15 to 34 we define the properties of the classes.One can see that for a dish in a restaurant (line 27-30) the object-resource has to be of type `http://www.food.com/ontology#dish` (line 29) which is also the class for dishes from Shah (see Listing 1.

By expressing the ontology in RDF, it becomes machine processable like the information about the restaurant Shah and its dishes. Note that in the listing above, horsepower is not defined for a restaurant. Also, for the property "numberOfTables" only non-negative integers are allowed, which is expressed in line 25. If now a software agent encounters a description which assigns the property horsepower to a restaurant or which states that a restaurant has "potato" as number of tables, the agent can compare it to the ontology and infers that the statements are not valid. This way, the agent gets a concepts for different entities and can process knowledge about them automatically.

Often, it is the case that rdf statements are not stored in rdf files, but in a database. These kind of databases are called triple store and can hold billions of statements. In order to easily retrieve the statements that are needed, SPARQL[49] has been developed by W3.[50]

SPARQL is an RDF query language language. It differentiates between four different query forms [66, p. 249]:

- SELECT-Query: For extracting values from SPARQL-Endpoints, which is presented in table format, similar to SQL.
- CONSTRUCT-Query: To extract information from a SPARQL-Endpoint which is translated into valid RDF.
- ASK-Query: Returns simple True/False results for a query
- DESCRIBE-Query: Returns an RDF Graph, which content is left to the endpoint and what the maintainer deems as useful information.

For our purposes, we might want to construct a select query that asks for a list of all dishes that are available for restaurants with less than twenty table in our vicinity.[51] Listing 4 describes such a query:

```
1 PREFIX RDF: <http://www.w3.org/1999/02/22-RDF-syntax-ns#>
2 PREFIX RDFs: <http://www.w3.org/2000/01/RDF-schema#>
3 PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
4 PREFIX est: <http://establishments.com/ontology#>
```

[49] An recursive acronym for SPARQL and RDF Query Language

[50] W3C Recommendation: `http://www.w3.org/TR/rdf-sparql-query/`

[51] We could also direct filter out all dishes that are not vegetarian with one query, but this would require to extend the request for multiple graphs, which is a too complex example

```
5  Select ?dish From <http://establishments.com>
6    where{
7           ?res a est:restaurant.
8           ?res est:address ?address.
9           ?res geo:geometry ?geo
10          ?res est:numberOfTables ?tables.
11          ?res est:served_dish ?dish.
12          Filter(
13            <bif:st_intersects> (?geo, <bif:st_point> ( 8.654,49.877),1.0) &&
14            ?tables < 20
15            ).
16         };
```

Listing 4: SPARQL-Query

The first four lines describe abbreviations for ontologies that will be used. The prefixed will be substituted with the URI part they where defined for.
The fourth line describe that data should be returned (the variable ?dish i.e. all addresses of the restaurants) from which graph (i.e. assembly of statements) we get the information(the one that is available at "<http://establishments.com>"). Line five to 14 define the a graph pattern for our query. This part defines variables and in which relationships they are to each other. Line six states that we want those resources that are of type restaurant and assign them the variable name "?res". Line seven state that of such a resource, its address has the variable "?address". We do the same for the number of tables and the served dishes in line eight and nine.
In line ten, we filter out all resources that do not match for the requirements "in our vicinity" and "with less than 20 tables". Line 11 is a custom extension from a SPARQL database which enables us to search for resources whose coordinates are in an 1km area around our defined spatial point. Line twelve states that we just want to have dishes from restaurants that have less than twenty tables.
Sending such a request to the SPARQL-Endpoint(i.e. the web address of an triple store) we would get a list of all dishes that are available in our vicinity.

2.3.2 Semantic Web and Data Mining

The same power of the Semantic Web that enables our agent to easily gather knowledge about entities and also to infer new knowledge about them. Such additional features, describing an entity, then can be used in machine learning applications, for example content-based recommender systems.
Finding such features describing is a task of Data Mining. This task can be easily simplified with a working Semantic Web. [14, p. 11f.] Suppose we want to find and integrate all information available about restaurants in Sunnyvale for a content-based recommender system. Without a Semantic Web, this would mean that first, we somehow have to identify all data sources that contain relevant information. After sources have been searched and identified, the next task would be to extract relevant information. This usually means that for every source, we have to create a crawler, tailored to the particular source, for extracting information we identified. As usually for every data source an own crawler has to be developed, this become an time consuming task. Such specialized crawlers are also prone to break if the underlying source changes, making them pricey to adapt.
With a working Semantic Web, both tasks of finding and integrating information are simplified tremendously. Suppose we have a list of all sources that can possibly hold information that are interesting to us, for example addresses of their SPARQL endpoints. Also, we have the URIs of all the restaurant we search information for. With a Semantic Web, we could simply request from all source all rdf statements they have, which contain one of the URIs. Such sources then would return all the statements that contain the URI. Then for every restaurant, we would have a set of statements, describing the resources we want to have information from. This would not make it necessary to manually check for available information and extracting them through a custom made crawler, saving a lot of effort.
Going even a step further, one could imagine a Semantic Search Engine, i.e. a Search Engine that indexes

all rdf statements available. In this case, we would not even a list of potential sources, but could simply query this search engine for information of the restaurants by naming the URI. Again, as a result, we would instantly have a set of rdf statements, containing information about the restaurants.

As already mentioned, with Semantic Web information about entities are not only obtained easily, but also new knowledge about them could be generated. Take for example Listing 1, that contained information about restaurant "Shah". This listing also contains statements that state which dishes are served at the restaurant. Based on the statement that link the restaurant to its dishes, one is able to generate new knowledge about the restaurant. For example, one could generate a feature that describes if Shah serves vegetarian dishes or the quota of vegetarian dishes. For a recommender, this information can be interesting if it turns out that users have preferences for restaurants that serve vegetarian dishes only. For extracting such information, an intelligent feature mining agent could traverse the list of dishes of a restaurant, and based on the question whether they are vegetarian or note compute a quota of vegetarian dishes. If a dish also holds the ingredient, one could generate features describing the restaurant based on the used ingredients, for example stating which sort of ingredients occur frequently or infrequently. For a user that has a strong preferences for lentil dishes, a recommender can then recommend a restaurant based on the ingredients of dishes served.
As it is costly to program such a feature mining agent for all potential objects that can be part of an rdf statement with a URI of a restaurant, i.e. programing how it should handle dishes and ingredients. Instead, by using heuristics, features describing a POI could be created automatically, for example with a rule such as "if there is a statement with a the searched resource as subject and another resource as object, and this object has a list of integer values, compute the average of these integer values and at it as a new feature to the subject", which would take the prices of all dishes served in a restaurant and compute the average, describing the pricing of the restaurant. By simply exploring the properties of the resource (and consequently the properties of the resources the initial resource is linked to), the agent automatically adds new information about the restaurant, without the need to manually identify information and assembling from multiple sources.

2.4 Related Work

In the following, research related to this thesis is discussed.
The first part presents research that qualitatively examines how and why users use LSNs and quantitatively describes patterns that unveil user behaviors.
Following this, research on recommender systems in generally and research on recommender systems for LSNs will be discussed, following an overview over approaches for context recommendation. In the final section, approaches similar to ours, where recommending systems are combined with semantic technologies, are discussed.

2.4.1 Research on location-based social networks

Research on LSNs can be classified into two categories. Qualitative approaches aim at investigating the motives and behaviors of individuals when using LSNs. Quantitative approaches investigate properties of a LSN's user population and its general usage patterns through large samples obtained from LSNs, using statistic means.

For qualitative studies, important research questions are the motives for people using LSNs, whose friend requests they accept, the user's check-in behavior and the rational behind it and what measurements users take to protect their privacy.
The research indicates that motives are to let other persons know where oneself is and to know where other persons are, especially close peers and family. [27] Other reasons are finding new and interesting places and people and "playing" LSNs, i.e. using earning achievements. [27] [41] For the first motive, Cramer et al. differentiates purpose-driven usage and social- and identity-driven usage. The first serves

as a way to simplify communication between users, as the position can be shared easily. For example in [27], users state that they share their location with their parents so they are less worried. Social- and identity-driven usage on the other side has the purpose of sharing lifestyles and information that enhances one's self-presentation. [22] Closely connected to this is the question who is on one user's friend list, i.e. who can see check-ins. In general, friend of one person are accepted as friends, but to some extend also co-workers and to an lesser extend family. At the same time, non-desirable friend request come from people that are unknown, but also people from workplace and family are considered people that are undesirable as friends, showing that friends are in generally desirable as friends, while the desirability of other groups is not that ambiguous. [22] The sharing behavior, a second question investigated, aims at unveiling when and why users check-in or chose not to do so. This behavior is potentially connected to the purpose of usage, as the surveys indicate that most users tend to not check-in at routine locations like school, workplace and home, as such places are considered "boring". Also, people declare not to check-in at places they do not want to be connected with (for example doctors or fast food restaurant), for reasons of shaping the image of oneself regarding ones friends. the gaming aspect seems to influence peoples check-in behavior too as users who "play" LSNs try to gain achievements and badges and thus shape their behavior in that direction. [41]

A second strand of research on LSNs aim at unveiling usage-patterns of LSNs through quantitative means. The data used is obtained by performingg a crawl on the data of an LSN to obtain usage data such as check-ins, friend-connections and profile pages.
Scellato et al. measure user activity on LSNs, based on a complete crawl of Gowalla in 2010. They analyze the number of friends, number of check-ins and visited places of Gowalla users. On insight that it is easier to accumulate friends than it is to accumulate new places and check-ins, possibly that the later two require to actually go to a place while the first only requires to accept a friend request. The fact that the number of new check-ins decays over time seems to indicate that user involvement tend to slows down when a user spends more time using the service. [55] Scellato et al. [57] [56] also investigated the connection between friendship of users and their location.
They determined the approximate home location of a user through his check-in behavior(for example defining the home location as the spot with the most check-ins) and analyzed the friend-ties with respect to the locations of the friends. A main finding in this research is that friendships between two users are more probable if the user's geographical locations are close to each other and that users with more friends tend to have larger distance between them and their friend's geographical location.
Some studies examine the temporal properties of check-in patterns [64], [44]. Their findings show that the general check-in activity pattern differs between weekdays and weekends. During weekdays, there are two peaks in the number of check-ins at 10:00 and 20:00, which resembles the beginning of work and the beginning of after-work activities. On Weekend days, there is a considerably different pattern, where the number of check-ins increases from 8:00 two 14:00 and remains constant till 24:00. The number of check-ins also depend on the category of the venue. For example, venues with the tag "cocktail" have the highest numbers of check-ins after 18:00 while check-ins for venues like "college" peak in the morning and slowly degrade over the day. Not only for the time, but also for the day of the week, variances in the check-in pattern can be observed with respect to venue categories. The number of check-ins for venues like college are considerably lower at weekend days, probably as no courses take place during this time. On the other sides, check-ins for venues like clubs are most frequently in the late hours of Friday and Saturday, i.e. when the following day is not a regular working day. Noulas et al. also observe the transition patterns between categories of locations, showing that fr some categories, there exist patterns between the location visited and the location visited before.
Li et al. collected data from Brightkite and analyzing users, movement patterns and social graphs. By clustering a user's check-in locations depending on the spatial location, they where able to distinguish four types of users: non mobile "home" users with one single geographical cluster, home-vacation users, with one big cluster and a significant amount of check-ins from numerous different clusters, home-work

users with two main clusters and other users, who do not fit into the other patterns. Based on the activity in the LSN, such as the total distance between check-ins, the check-ins and friends, they further identified five user groups, from users who just try the network out (41&) and inactive users(30%) to normal users(16%) active(6%) and mobile users(8%), the last group being a group with high distance between check-ins. [39]

Another field of study occupies itself with modeling user's check-in behavior for predicting user's location. Backstrom et al. model a user's position based on his friend's position. They use a probabilistic model that determines the probability of a user being at a certain position, based on the distance of the position to all his friends whose position is known. For simplicity, they assume that a user will be in proximity of one of his friends. Using data from their employer Facebook, their results show that under certain conditions regarding the number of friends and their distance, this method can yield better results than IP-based geolocation. [10] Another model is developed by Gao et al. They show that users tend to go to the same places their friends go and that the check-in can be described by a process incorporating the user's previous check-ins, where older check-ins have less importance than newer. Based on these two observations, they develop a model only based on the check-in sequence as well as a model based on the check-in sequence and the user's friendships. When evaluating, these models show a higher accuracy for predicting a user's next check-in than for example models that only rely on the frequency of check-ins. [28]

2.4.2 Recommender Systems

Research on recommendation systems has been performed for some time for example for e-commerce applications, resulting in in a rich literature on this topic. For an overview over different approaches and the most important implementations, one can refer to Adomavicius et al. [2],Ricci et al. [53] and Brusilovsky et al. [20].
Research on recommendation related to LSNs, in comparison, is still relative sparse, as LSNs itself are a relative new topic.
Prior to the emergence of LSNs, research in the area of POI recommendation was performed for mobile tourist guides, for example as part of the "COMPASS"-tourist application. [59] The data in such studies was usually obtained with test users, for example by handing out GPS transceivers to test subjects or by creating catalogs of POIs where users could vote for different POIs, such as restaurants. For example Horozov et al. created a system for restaurant recommendation, where test users could vote for restaurants and receive recommendation. For recommendation they used an collaborative approach. First, a list of POIs in in the surrounding of the user is computed. For collaborative filtering, all users who casted a vote on these POIs are considered, taking the N most similar of them to derive a rating. [33]

Research on recommendation for LSN on the other side is based on historical data that has been crawled from a LSN. The research here focuses on lowering the computational requirements for collaborative filtering by exploiting the spatial component of a user's check-in as well as the properties of the user's social network.
Agrawal et al. propose an approach called geo-measured, friend-based collaborative filtering for lowering the computational overhead. They observe that friends tend to share more locations than two random non-befriended users. This is especially true if the friends often check-in in each other's proximity. They develop a friend-based collaborative filter approach where the similarity of two friends is weighted with the distance of spots the spots where the two checked-in. Their approach is less effective but much more efficient compared to approaches like normal collaborative filtering or friend-based collaborative filtering. [42]
Gupta et al. try to lower the costs for collaborative filtering by focusing on users that have the same spatial affinity like the user as the basis for similarity measures. They develop two metrics to form regions, using them to describe the area where a user is active. Then, similarity of two users is only computed if

their overlapping activity area is sufficiently high. They evaluate this approach using a synthetic dataset that simulates a LSN and show that the focus on users with similar regions has little negative impact on quality and coverage while decreasing the time taken to make a recommendation. [29]
Mai et al. use a hybrid approach for recommending POI to users where three different recommenders are combined. The first is user-based collaborative filter, where a number of similar users, measured by their check-in behavior, recommend spots to a user. The second approach is friend-based collaborative filtering, where the N most similar friends with respect to their behavior are used to recommend POIs. A third approach uses POIs in the area of a POI to be rated that have been visited by the user before as basis for deriving a rating. Those three approaches are combined using a linear model, where the rating of a POI is the weighted mean of the three recommender systems. Their findings show good results for their unified recommender compared to the isolated approaches. Interestingly, the third recommender, using geographical proximity of other spots, had a more significant impact than social influence from other users. [65]

Berjani et al. implement a recommendation model-based collaborative filtering approach. For this they crawl a dataset from Gowalla including user profiles, spot profiles and check-in data. They use a method called equal width intervals to achieve a discrete rating for spots by users, where the rating is computed from implicit observed check-in frequency by dividing the maximum number of check-ins minus the minimum number of check-ins through the number of check-ins. Based on this, they use regularized matrix factorization, which is an approximation technique for collaborative filtering, to recommend items to users. Their implementation shows a better performance than an average item recommender. [15]

Research on incorporating context in recommendation systems, as already mentioned in section Section 2.2.4 can be divided into approaches using pre-filtering, post-filtering or context-modeling. Most approaches encountered in the literature are pre-filtering approaches, where data is adopted to the context prior recommending items or to building a recommendation model.
Baltrunas et al. follow such an approach, which they call user splitting. They create several, possibly overlapping, user profiles with a model based collaborative filtering approach for recommending music, considering the context of time. The models are build using only data from the context and evaluated only with data from the corresponding context. In a first experiment, they try different time segmentations, namely morning or evening, if the day is a weekend or a working day, if the season is cold or hot and if the hour is even or odd. Here it shows that some of these profile segmentations have better results than the full user profile. In a second experiment, they split a day into two 12 hour profiles, varying the splitting point. Here, their results show that the choice of the right point can be of importance, as the mean absolute error of recommendation varies significantly, depending on the split point. [12]
Adomavicius et al. propose an approach called generalized pre-filtering. They understand context as a n-dimensional space which can be partitioned into segments, for example by segmenting the dimensions of day and place of watching a movie into the segments (weekday,home), (weekend,cinema), (weekday,home) and (weekday,cinema). In general, such segments can also be a subset of other, conceptually more broader, segments. By either pre-defining segments by an expert or considering every possible segment, they get a list of segments they start with. Now, using test data, every possible segment is evaluated for itself, using a traditional 2D method, comparing it to a global "traditional" 2D recommender scheme. A segment is rejected if it shows inferior performance compared to the global model. Additionally, of the remaining segments, also those are rejected that are a subset of a segment whose performance is superior. Using this approach, they get a list of high performing segments. For recommending items given a context, the segment is used for recommendation which showed the best results during evaluation for the context. If a recommendation situation is not part of any of the evaluated segments, a global 2D recommender is applied. [3] Ricci et al. perform an approach called item-splitting, where an item is split into multiple "virtual" items, each one being the initial item under a different context. This can

also be considered pre-filtering, as the rating matrix is adjusted using contextual information prior to recommendation. Items are split in two different items, for example a movie on weekday and a movie and weekend) if the rating between those items differ significantly. While this method may lead to more accurate recommendation, the downside is that the sparsity of a rating matrix is increased, as for the same number of users and ratings, more items are available. [13]
Cremonesi et al. use association rules in a post-filtering approach for movie recommendation. After a collaborative recommender system generates a list of recommendations, a subset of these items is selected that relate to the context according to the rules, which is then presented to the user. Rule mining is performed for the context age, gender, occupation and zip code of the users. This methods allows to increase the recall for the topN list of their recommender system. [46]
Panniello et al. experimentally compared pre-filtering with two different post-filtering approaches. Pre-filtering is done by only considering ratings that where given under a certain context. For post-filtering, they compute a contextual probability. For a user and an item given a context, the probability is defined as the number of users similar to the given user (assessed by the similarity of users) divided through all users who purchased the item given the context. This probability is then either used for weighting ratings by multiplying them with the probability or by filtering them, where the rating is set to zero if the probability is lower than a given threshold.
Their results show that pre- and post-filtering methods can lead to better results. In their experiments, post-filtering with filtering out items with a low probability generally has better results than weighting the scores with the probability and that pre-filtering is sometimes better than an un-contextual method. They not though that comparing different approaches for the application at hand is important to identify the proper contextual strategy.
Lastly, Oku et al. followed the approach of contextual modeling, extending an SVM to incorporate contextual dimensions for restaurant recommendation. Used contexts are the time, situation variables including the users budget, the type of area and if it is a holiday, the partners who are with the users as well as external factors, such as weather and temperature. The thereby enhanced SVM-based collaborative filter approach was then evaluated in a small scale experiment with a low number of users and restaurants. Their results suggest that the integration of context can help to enhance the accuracy of the recommender and user's satisfaction with the recommendations. [45]

2.4.3 Semantic Recommendation

There has also been some research on recommendation systems using semantic technologies. Such approaches work by exploiting the ontological organization of items, the links between entities or both.
This is turn can be used directly for content-based recommendation or serve as basis for collaborative filtering, where the similar rating of two users is not just computed based on their rating of items, but also on ratings of classes of items.

Some research has been done in the area of creating ontologies for existing items to be recommended. This hierarchical ordering of items can be used to compute the similarity of items by their position in the ontological tree. This in turn can be used to compute the similarity between users.
For example Wang and Kong use such an approach for a personalized recommender system. For calculating similarity between users, they first create an ontology to categorize the items to be recommended. Then, they compute item similarity on the basis of the ratio of common shared classes. This similarity is then combined with the a traditional user similarity measure using Pearson Correlation Coefficient and measure of users similarity based on demographic attributes into an similarity measure for users. With this similarity measure, a model-based collaborative filtering approach, clustering users into groups is implemented. [62]
Kim et al. create an ontology for a grocery store with products, records(transactions), consumers and the contextual dimension of location. The preferred items of users are computed based on the item he has purchased as well as the classes of items he has purchased. If a user buys an item, he shows pref-

erence not only for the item itself, but also for all classes of items along the ontological tree up to the root. Depending on the specificness of the context chosen by the user (i.e. its specificness of location), recommendations on items can be more or less specific (i.e. more up or down in the hierarchical product ontology). [36]
Another strand uses existing semantic data from linked data projects to recommend items. For their approach, they not just use the ontological hierarchy of items, but also information available from entities linked to an item through RDF statements.
For example Di Noia et al. developed a content-based recommender system solely based on data from DBPedia, LinkedMDB and Freebase. First, they transformed the RDF graphs into a vector space model where two axis hold entities while the third holds properties. If two entities are connected through a property, the corresponding coordinate in the vector space is not null. This matrix can be further decomposed into matrices for every property, with one axis denoting the domain (the objects of the statements of this property) and the other the range on values (the subjects of the statements of this property). Based on this, the similarity of items can be computed using cosine similarity in these matrices, combining them into one similarity measure. For evaluation, they considered different definitions of when an item is connected through a statement to another subject. Besides only using direct links, they also consider transitive connections, meaning that two objects are connected if they are connected through multiple nodes. Their results show that by broadening the available properties, also from different databases, an improvement in performance can be achieved. They note that additional information can harm performance if too "distance" information, are added, as this adds more generic information. It turns out that the information that are at most one "step" away from the items to be recommended offers the highest benefit for recommendation. [24] Likewise, Passant uses entries for DBPedia, a RDF-ization of Wikipedia entries, for music recommendation. They define a distance metric on the basis of linked data by incorporating the number of direct links between to entities as well as the number of indirect links over one other entity between two entities. This approach ignores object hierarchies, as it only operates on the graph of entities to be recommended linked directly or indirectly to each other. Using these information in a distance metric, they compute for every artist recommendations in the form of similar artists, where smaller distance means that the two artists are more similar. They evaluated their experiment using an small scale laboratory experiment where user stated their favorite bands and the resulting recommendations where given them for evaluation. [48]

3 Datasets

This section presents data sets and sources used in this thesis.
The first section describes a dataset from Gowalla, which serves as basis for our recommender system. After a short introduction of Gowalla's history and its core features, we present the structure and content of the data set, following a description of its key properties, including activity patterns of users and spatial spot and check-in densities.
The second part deals with possible knowledge sources form the Cloud of Linked Data that we can use to enrich our dataset. We evaluate the general situation in the Cloud of Linked Data regarding the availability and usefulness of sources for our purpose. Although most of them are of little use to us, two sources, LinkedGeoData and MesoWest, appear suitable for enhancing existing Gowalla data. These sources are introduced in the last two sections of the second part and for each, a description of its background and key properties is provided.

3.1 Gowalla Dataset

Gowalla was a location-based social network, founded in 2007 in Austin, Texas and launched in 2009. Facebook acquired Gowalla in 2012[52] and announced it would integrate it into their service.[53] By 2012, the website of Gowalla had been closed and the service was no longer available.[54] Gowalla was one of the "pure" LSNs, whose core-feature were location-based services. Users could become friends with each other, realized as a two sided-connection, and perform check-ins at POIs called "spots". These spots where created by users, giving them a name, a position and additional information such as a description or a categorization. Check-ins could be performed by a user if he was within a predefined range of a spot. The information about the check-ins were shared with other friends and could be seen on the profile page of the spot the user checked-in.
Gowalla also featured the ability to gain achievements. For every spot visited, a user received a stamp of the spot. Also, a user received badges for performing certain activities, for example when performing a check-in at a certain number of locations or when he created a spot.

The data used in this thesis was acquired by crawling the Gowalla API from June to October 2010 as part of another master thesis conducted by Betim Berjani [17]. Additional crawling has been taken place to further enhance the data.
Figure 2 illustrates the data model for a user. Central for identifying a user is his userID. Table "gw_user" holds information that connects a userID to a loginNr and a synonymLogin, which were Gowalla's keys for identifying users, the date when the user information was crawled for the first time.[55] Table "gw_userFields" holds more detailed information about a user. The information available are those which where visible on a user's profile page, and where either provided by the user itself, such as his first and last name, hometown, a short biography or the URL of his website and his Twitter or Facebook account, or aggregated statistical values, such as the number of performed check-ins, number of acquired stamps or items and the number of confirmed friends.
For a single user, multiple entries with different userFieldIDs reflect state of the profile information for different times when the information were crawled, enabling to track changes in the profile information over time.
Table "gw_friend" holds a list of connections of the user to his confirmed friends. As a friend connection is coupled to a userFieldsID, it is possible to track changes in the social graph of users.

Figure 3 shows the data model of a spot. Table "gw_spot" with the primary key spotID, serves as a table

[52] http://techcrunch.com/2011/12/02/report-facebook-has-acquired-gowalla/
[53] http://blog.gowalla.com/post/13782997303/gowalla-going-to-facebook
[54] https://mashable.com/2012/03/11/gowalla-shuts-down/
[55] The userID was introduced by Berjani for integrating all user information and was not the unique identifier of Gowalla

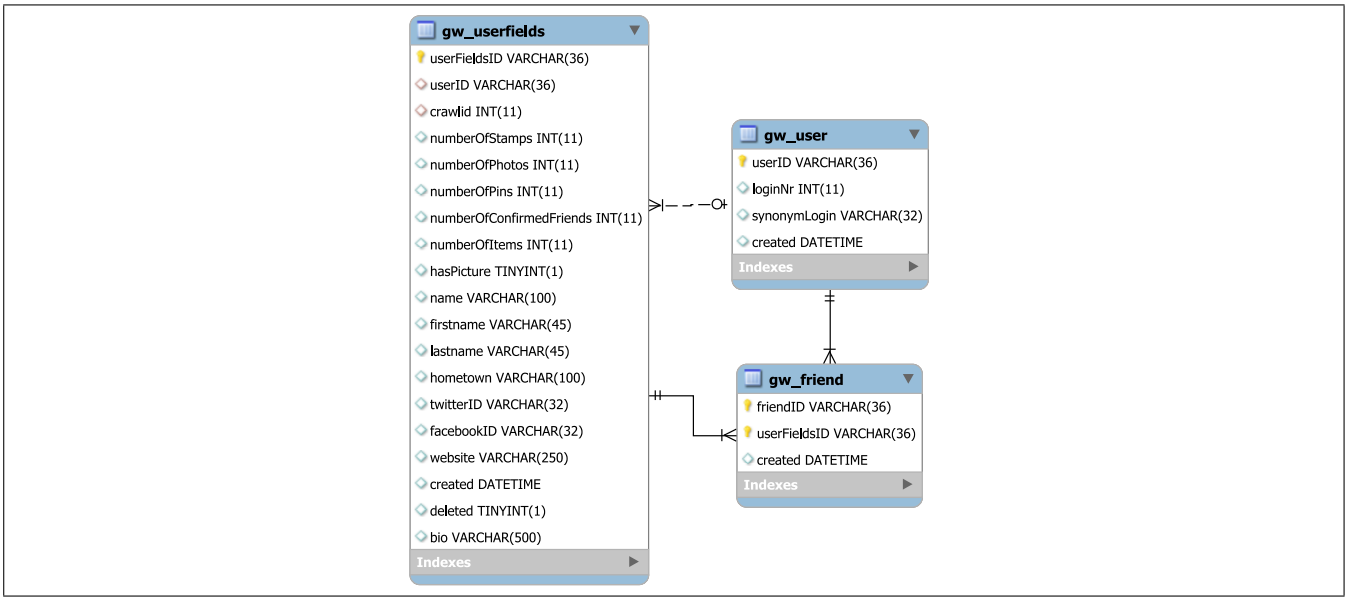


Figure 2: Data model of users in Gowalla dataset

for combining the spotID and his first crawl date. Similar to users, table “gw_spotFields” enables to hold different versions of information on a single spot, enabling to track changes over time.
Table “gw_spotFields” holds various information that where visible on a spots’ profile page. They were supplied with the creation of the spot, but could be altered afterwards. Entries in this table include a userID of the spot’s creator and its creation date, the name of the spot, a locality (which is usually a city) and a region (which is usually a state), a description of the spot by the creating user and a twitterID and website associated with the spot. Similar to a user’s entry, a spot entry has some aggregated statistics, such as number of performed check-ins, the number of photos taken at the spot and the number of unique visitors. The spatial information about a spot consist of coordinates in latitude and longitude and a radius, which is used for deciding if a user is in the vicinity of a spot and therefore near enough to check-in.
Spots may also have a category assigned, identified through a categoryID at table “gw_spotcategory”, while table “gw_category” holds information (name and creation date) of the category itself.

The data model for check-ins can be seen in Figure 4. The event of a check-in, i.e. a user checking-in with his mobile device into a spot, is represented as an entry in table “gw_checkin”, which incorporates the spotID of the spot where the check-in occurred and the userID of the user who checked-in. It also includes additional information related to the check-in, such as the time of check-in, comments or an indicator if a picture is connected to it.

3.1.1 Characteristics of the Gowalla dataset

This section describes key properties of the aqcuired Gowalla dataset. Table 4 provides an overview over the number of entries for the mentioned tables.
In total, around 1,85 million spot informations were crawled. The number of entries in “gw_spotFields” with around two million higher, as it also holds the spots’ change histories. Of the 1,885,742 entries in “gw_spot”, 1,722,844 (91%) have corresponding entries in “gw_spotfields”. Of those spots, 83% have never been changed after creation, 17% have changed once after creation and less than 1% have been changed more than once. Although for 9% of the spots no profiles are available, their check-in informations are, making up 8.5% of all check-ins performed.
From all spots, 73% have a category assigned. The biggest categories are “Gas & Automotive” (4%), “Asian Food” (3%), “Corporate Office”(3%), “Other” (2%) and “Grocery” (2%).

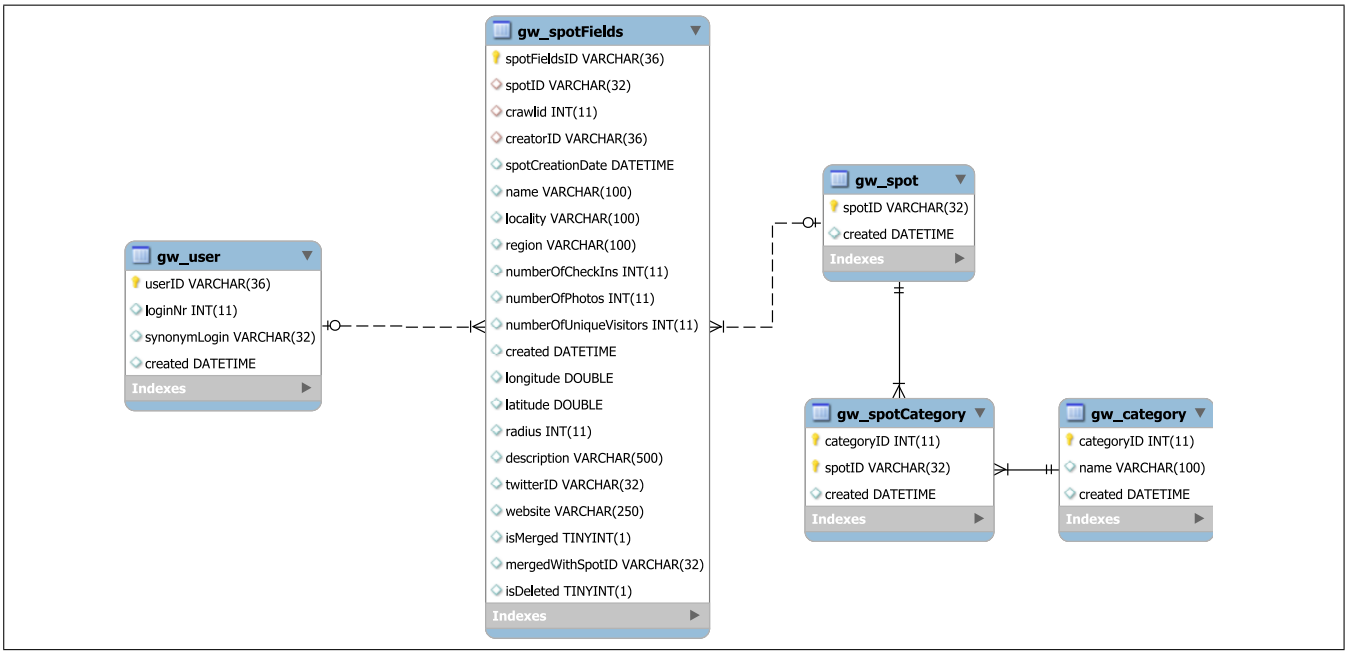


Figure 3: Data model of spots in Gowalla dataset

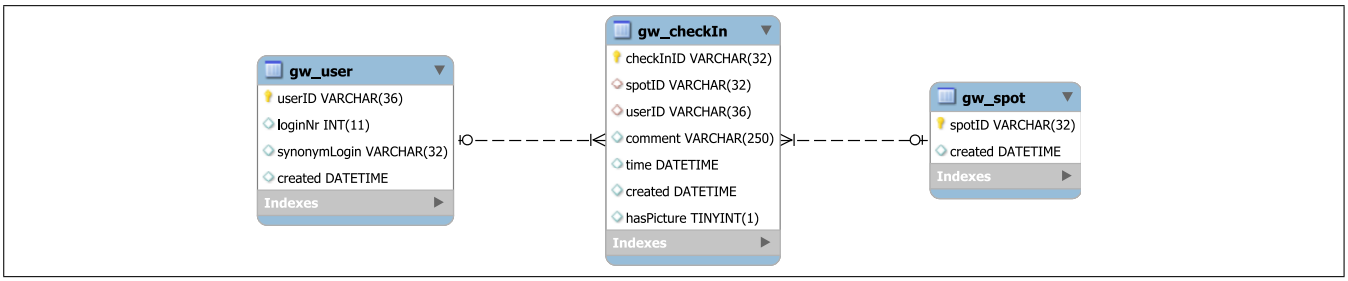
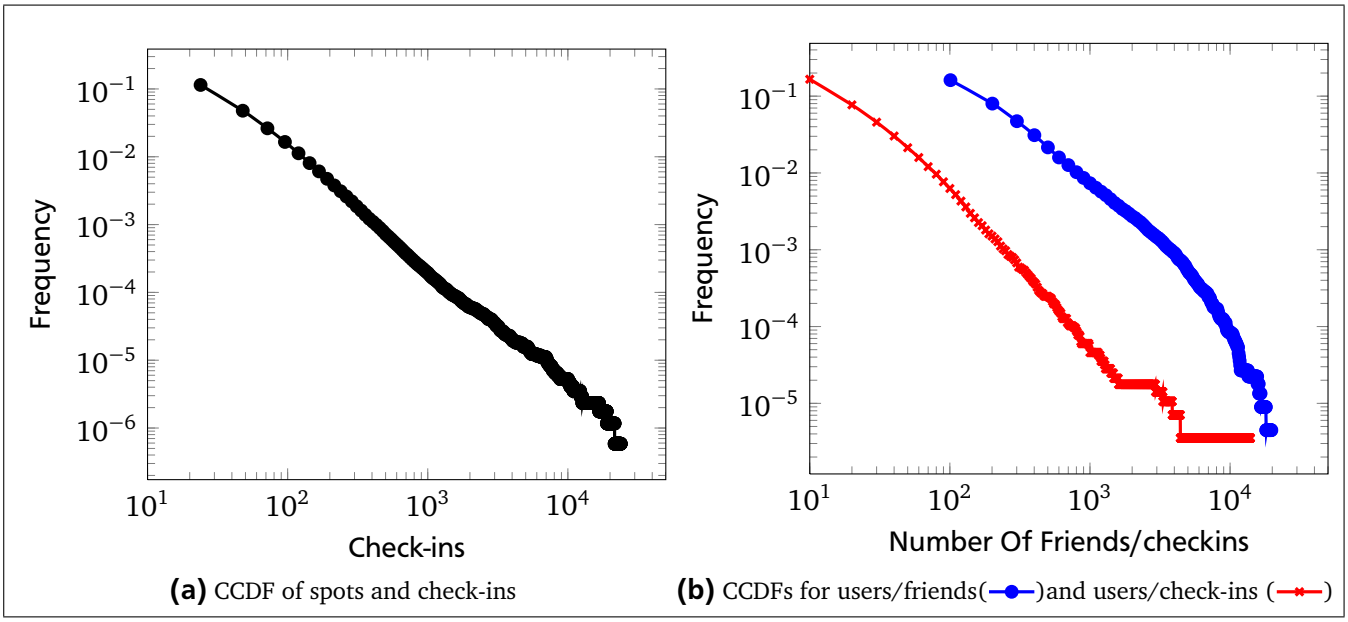


Figure 4: Data model for check-ins in Gowalla dataset

Table	# Entries
gw_spot	1,885,742
gw_spotfields	2,026,172
gw_spotcategory	1,371,399
gw_categories	457
gw_user	259,650
gw_userFields	290,558
gw_checkin	16,564,034
gw_friend	1,075,293

Table 4: Sizes of different database tables form the Gowalla dataset

On the other side, nearly 260 thousand user information have been crawled and for 177,113 of them, profile information is available. Of these users, 38% have no change of their profile page recorded, 58.61% have one change, 2.68% two and the rest four times. Similar to spot entries, for users where no entries in "gw_userFields", check-in information are available, which make up around 10% of all check-ins.
Complementary cumulative distribution functions (CCDF) for the check-in of users and the number of friends are presented in Figure 5b. Both show a heavy tail reported in other studies.([44], [57], [56]) Figure 5a on the other side show the CCDF for spots and check-ins. Also here, the distribution shows a heavy tail, with only eleven percent of all spots having more than 23 check-ins.
Another interesting aspect is the usage of Gowalla in different countries. To examine this, Table 5 shows



(a) CCDF of spots and check-ins **(b)** CCDFs for users/friends(—●—)and users/check-ins (—×—)

Figure 5: CCDFs for spots/check-ins, users/friends and users/check-ins

the share of spots and check-ins for countries with a share higher than 1% of both as well as the nubmer of spots and check-ins relative to the countrie's population.
As can be seen, most of the countries (13 of 15) are Western countries, with two Asian countries (Taiwan and Japan) and one Arabic (Saudi Arabia). With nearly half of all spots and more than half of all check-ins, Gowalla seems to be most active in the United States. This may be due to the fact that Gowalla is an U.S.-based company. Other countries with major shares of spots and check-ins are mainly western countries, including Sweden, Germany, United Kingdom and Canada. Exceptions are Thailand, with 2.3% of all spots and 3.39% of all check-ins and Saudi Arabia with 1.53% of all spots and 2.45% of all check-ins.
With a look at the numbers for spots per capita and check-ins per capita, it first occurs that Sweden, which is placed second in absolute numbers, has by far the highest values, with more than six times more spots per capita and more than seven times check-ins per capita that the U.S.. Only Australia has similar values, though it has far lower share of spots and check-ins. Germany, United Kingdom and Canada, placed third to fifth, have roughly the same spots per capita and check-ins per capita, with roughly a half of spots per capita and forty percent of check-ins per capita compare to the U.S..

Figure 24 in the appendix shows the distribution of spots on a global heat map. Note that in general, major concentrations of points are in cities. In the US, those include San Francisco, Los Angeles, New York, Washington, Dallas, Houston and Austin. The latter three might be due to the fact that Gowalla resided in the same state and had its headquarter in Austin.

Also note that on the European continent, the place with the highest density is Stockholm, the capital of Sweden, but also other major cities in Sweden, Gothenburg and Malmö, show a high density. As 39% of the swedish population lives in this cities and Sweden has an overproportional number spots per capita, it comes to no surprise that these cities have the highest densities of spots. Given that the clusters are comparable in their size to for example London, the importance of Sweden becomes even more obvious if one takes a look at the population numbers of those cities, which are considerably lower.[56].
Germany, the Netherlands and Belgium on the other side, do not show such strong concentration of spots but larger areas of moderate density. Those include Berlin, the Rhein-Main-Region, Munich as well as the Rhine-Ruhr Metropolitan area in Germany, the Antwerpen area in Belgium and the Amsterdam area in the Netherlands. For England, there are major concentrations for London itself as well as Manchester. For the rest of Europe and the world, the density tends to be focused on major cities. A prominent example is France, where only Paris seems to be an area of higher density. Other cities with major clusters of spots include Tokyo, Bangkok, Riyadh, Seoul and Sydney. Of course, these numbers always have to be seen in comparison to the population.[57]

Next, Figure 25 in the appendix shows a heat map of check-in densities. Compared to the spot densities, the check-in activity is considerably lower than the spot density for some regions, for example larger cities of Australia, many cities in the U.S. or areas in Central Europe.
Activity seems to be concentrated on fewer cities, including San Francisco, Austin, Los Angeles and New York for the U.S. and London the aforementioned Swedish cities Gothenburg Malmö and Stockholm. Also note the very high activity in Stockholm, which sets itself even more apart from other cities. Outside Europe and North America, again Riyadh and Bangkok are additional hot spots of check-in activity.

These results lead to the conclusion that the distribution of spots and check-ins is quite uneven, with North America and Europe being the center of Gowalla's activity.

Country	% Spots	% Check-ins	spot per capita	check-ins per capita
United States	48.41%	54.61%	0.0031	0.0469
Sweden	9.11%	11.77%	0.0192	0.3352
Germany	6.60%	5.65%	0.0015	0.0176
United Kingdom	5.09%	4.11%	0.0016	0.0176
Canada	2.61%	1.75%	0.0015	0.0139
Thailand	2.34%	3.39%	0.0007	0.0138
Norway	2.18%	2.51%	0.0090	0.1395
Belgium	1.61%	1.20%	0.0030	0.0297
Saudi Arabia	1.53%	2.45%	0.0124	0.2674
Australia	1.50%	1.00%	0.0373	0.3345
Netherlands	1.45%	1.15%	0.0170	0.1818
Spain	1.26%	0.53%	0.0006	0.0031
Japan	1.22%	0.74%	0.0002	0.0015
Italy	1.13%	0.44%	0.0037	0.0192
Switzerland	1.13%	1.06%	0.0029	0.0369

Table 5: Quota of spots, check-ins, spots per capita and check-ins per capita for countries with more than 1% of spots and check-ins

[56] Stockholm metro area: 2.1 million, Gothenburg metro area: 0.93 million, Malmö metro area: 0.6 million, London metro area: 13.70 million
[57] Tokyo metro area: 35 million, Bangkok metro area: 14.5 million, Riyad metro area: 6.8 million

3.2 Linked Data

As we would like to leverage the power of semantically annotated data for generating additional features for entities from Gowalla, data sources that are mainly interesting to use are those whose data is published using Semantic Web standards. A good place to begin the search for semantic data is the Linked Open Data Cloud. Linked Data is a method for publishing and interlinking structured data[31, p. 7]. The principles of Linked Data where formulated by Tim Berners-Lee as following[58]:

1. User URIs as names for things.
2. User HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
4. Include links to other URIs. so that they can discover more things.

The LOD Community project[59] aims at publishing datasets in accordance with these principles. The data published by this project and other individuals and organizations form the LOD cloud, a collection of nearly 300 datasets with over 31 billion triples as of September 2011.[60]
This section reviews available data from the Cloud of Linked Open Data for the inclusion to our existing Gowalla dataset. As location is the core concept of LSNs, we are mainly interested in data that has some spatial component. The concept of space has already gained some attention in the Semantic Web Community. In 2003, the W3C Semantic Web Interest Group[61] released a vocabulary[62] for representing latitude and longitude information. This was picked up by the W3C Geospatial Incubator Group[63], which presented two reports on a geospatial vocabulary[64] and a geospatial ontology[65] in order to enhance the initial vocabulary.
In the LOD-Cloud, spatial data is also an important categoriy. The report “The State of the LOD-Cloud”[66] from September 2011, which examines how the LOD-Cloud is composed at the time, lists 31 resources with around 6 billion triples in the "“Geographic"” domain. This constitutes around 20% of all data in the LOD-Cloud.

These 31 resources vary in its nature, availability and quality. Most sources are not suitable to be combined with our Gowalla dataset for various reasons.
First, not all listed resources seem to be available anymore, such as “Weather Stations”[67], a source that published weather reports from airports.
Second, most datasets do not seem to contain relevant information about entities from Gowalla. For example, the “European Nature Information System”[68] holds information about species, habitats and conservation areas. As these areas are usually not places where a LSN user would check-in, this knowledge seems not very useful to us. Similar examples are the “Ocean Drilling Codices”[69], which hold information about deep sea drillings or “Pleiades”[70], which holds data about ancient world studies.
But also if the data is of principal relevance, it might not hold relevant data for regions with high density

[58] http://www.w3.org/DesignIssues/LinkedData.html
[59] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
[60] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData#Project_Description
[61] http://www.w3.org/2003/01/geo/
[62] With the namespace URI http://www.w3.org/2003/01/geo/wgs84_pos#
[63] http://www.w3.org/2005/Incubator/geo/
[64] http://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/
[65] http://www.w3.org/2005/Incubator/geo/XGR-geo-ont-20071023/
[66] http://www4.wiwiss.fu-berlin.de/lodcloud/state/
[67] http://thedatahub.org/dataset/los_metar
[68] http://thedatahub.org/dataset/eunis
[69] http://thedatahub.org/dataset/oceandrilling-codices
[70] http://thedatahub.org/dataset/pleiades

of spots and time spans where check-ins occurred. Many sources are country-specific or even regional as they have been published by some national or regional agencies or programs, for example the open data project from the Italian region of Piemonte [71] Another example is the the dataset "Street level crime reports for England and Wales" by the School of Electronics and Computer Science and the University of Southampton, which offers crime reports in RDF. For this source, the earliest data available is from December 2010. But as most check-ins [72] . Integrating such data seems uneconomical, as data can only be added to a small number of spots that are in the region, or for a large number of check-ins data is unavailable due to time and space constraints.

The restrictions imposed for ensuring meaningfully data excludes most datasets for integration. Fromm all data sets, only two proved to be relevant, LinkedGeoData and MesoWest.
LinkedGeoData (LGD) .[73] offers information about POIs they acquired from OpenStreetMap[74], a collaborative project to create a free and editable map, published as Linked Data. These POI information can be used to add knowledge about the spots in Gowalla.
The second source, MesoWest, was discovered through a linked data project, MesoWest[75] is a project that offers weather data from stations located primary Northern America, with historic data dating back to 2002.It was discovered through a linked data project, called "Kno.e.sis"[76], which converted data from MesoWest into Semantic Data, but only for a limited number of regions and periods. MesoWest itself does not offer their data as Linked Data, but encoded in XML, available through an API.
The reasons why this data was used, despite its non-optimal format are as follows. As the data is available in XML and acquirable through an API, easy acquisition and processing is ensured.[77] Secondly, this data offers the opportunity to integrate the additional factor of "weather" to check-ins. When using information from LGD to enhance our knowledge about spots, MesoWest data can be used to enhance our knowledge about check-ins. This allows to enhance recommendation for POI through two paradigms, using both content-based and context-based approaches. In the following sections, LGD and MesoWest are described in more detail.

3.2.1 LinkedGeoData

LinkedGeoData is a project initiated by the research group at the Institute of Computer Science at University Leipzig. It provides an integrated and interlinked geographic dataset for the Semantic Web, based on data from the Open Street Map Project.
Open Street Map (OSM) is a collaborative project to create free editable maps. It follows the peer production model, where every (registered) user can add details, leading to an detailed community-created map. The reasons for participating in this kind of project, ranging from ideological views such as to use free information to improve the world, anti-national mapping agency views or for the sake of fun, to go out mapping or sitting at home and writing code or enjoying being part of a community.[30, p. 16f.]
The fundamental entities used in this project are nodes, which are geographic points, such as a town, a statue or a restaurant[78], ways, which are a list of nodes, describing something like roads or streams, and relations, where points, ways and other relations are related to each other, forming complex entities. For example the administrative boundary of Germany comprises more than 100 ways, combined into one relation.
By now, OSM offers 1.6 billion nodes, 1.5 billion ways and 1.5 million relations, contributed by 850

[71] http://dati.piemonte.it/
[72] http://crime.rkbexplorer.com/
[73] http://www.linkedgeodata.org
[74] http://www.openstreetmap.org/
[75] http://mesowest.utah.edu
[76] http://thedatahub.org/dataset/knoesis-linked-sensor-data
[77] According to the five-star rating of Tim Berners-Lee, this data would get a 3 out of 5 star rating for openness http://www.w3.org/DesignIssues/LinkedData.html
[78] Ignoring the circumference of such nodes.

million users. Apart from single user contribution, there have been imports of public domain or donated data sources, such as data describing global shorelines or houses in Rostock.[79]

Entities in OSM can be tagged to add information to them. For different nodes, ways and relations, there exists a wide variety of tags for describing them.[80] For every tag, there are usually informal recommendations how the tags should be used. For example, for node that has the tag ”restaurant” assigned, an additional tag “cuisine” is advised, to describe the cuisine of the restaurant. [81]

LGD imports these entities, heavily relying on the assigned tags to add information to the linked data. In their import process, the entities first undergo a filtering procedure based on black- and white listing of tags in order to keep only useful entities. Via rule-based filters, LGD then identifies three kind of tags, namely classification attributes, description attributes and data attributes. The first category of tags is used to create an ontology for classification of nodes, while the latter two are used for describing the entities through properties.[9]
Additionally to the tags, some nodes are automatically interlinked to various other data sources, including DBpedia[82] (a linked data version of Wikipedia data), GeoNames[83] (a RDF database of countries and administrative regions) and the Food and Agriculture Organization of the United Nations[84]. For DBpedia, this linking is performed for nodes that have a corresponding DBpedia entry. GeoNames provides names and alternative names for geographical features, such as countries and towns. The Food and Agriculture Organization supplies additional data about countries, such as currency, the languages or the gross domestic product.[60]
Their approach created a database of 65 million triples, corresponding to 6.2 million nodes and 66 million triples corresponding to 7.1 million ways. Of these nodes, largest classes[85] are “parking”(8.3%), “village”(8.2%), “shop”(7.9%), “hamlet”(6.6%), “school”(5.7%), “PlaceOfWorship”(5.7%), “Restaurant”(2.7%), “FastFood”(1%) and “Pub”(1%).[60, p. 14]

Table 6 shows an example of information available for an LGD node. From our experience, the extend of information is above average, as the available information include the address as well as the availability of certain beverages.
The first row indicates the type of the node. Here, café is a subtype of amenity, which in turn is a subtype of Node. The second row shows the direct-type, i.e. the type of the node that is a leaf in the ontological tree. The next line refers to the geometrical shape of the object, in this case a simple type. There exist some other types, such as polygons, but in our experience, they are rarely available. The next properties offer address information, such as the country the point is in, its city and postal code, as well as the street name and house number. Next, two properties describe the geographical coordinates, expressed as latitude and longitude. Following this, there is a property pointing to the URI of the user who lastly edited the entity, following the date it was modified.
Finally, there are additional information describing the node, namely, if it offers Club Mate and Afri Cola, if smoking if allowed and the wheelchair accessibility.

The data created is both available through an REST-API as well as through SPARQL endpoints. During the thesis, a static and a dynamic SPARQL endpoint was available. The static endpoint was created from an OSM obtained end of dating 2011. The dynamic endpoint regularly updated its data from OSM.

[79] See `https://wiki.openstreetmap.org/wiki/Import/Catalogue` for a list of sources
[80] A list of widely used features is given at `http://wiki.openstreetmap.org/wiki/Map_Features#Primary_features`, an overview about the tag usage is given at `http://taginfo.openstreetmap.org`
[81] http://wiki.openstreetmap.org/wiki/Key:cuisine
[82] `https://http://dbpedia.org`
[83] `https://www.geonames.org/`
[84] `http://www.fao.org`
[85] i.e. nodes with a certain tag

Property	Vale
rdf:type	http://linkedgeodata.org/ontology/Node http://linkedgeodata.org/ontology/Amenity http://linkedgeodata.org/ontology/Cafe
lgdo:directType	http://linkedgeodata.org/ontology/Cafe
geom:geometry	lgd-geom:node548217289
lgdo:hasCountry	DE
lgdo:hasCity	Darmstadt
lgdo:hasPostalCode	64289
lgdo:hasStreet	AlexanderstraSSe
lgdo:hasHouseNumber	2
geo:lat	49.875035
geo:long	8.6575656
lgdo:contributor	http://linkedgeodata.org/triplify/user290680
dcterms:modified	2011-02-13T11:50:34.0+01:00
lgdo:drink%3Aclub-mate	yes
lgdo:drink%3Aafri-cola	yes
lgdo:smoking	no
lgdo:wheelchair	http://linkedgeodata.org/ontology/limited_%28WheelChair%29

Table 6: Example of node information for node with ID 548217289 in LinkedGeoData

While conduction the thesis, we used the static endpoint, as the dynamic surprisingly offered less nodes and was prone to frequent downtimes.

Regarding the spatial pattern of nodes, one can refer to the Figure 26 in the appendix. The highest density of points is Europe, especially Germany. There is also a higher density of nodes in the U.S., Israel, an South Korea. The bright red dot on the border to Poland is the area around Szczecin. A closer investigation led to the high activity is due to the fact that in this region, a very high number of OSM-Nodes representing trees had been created, which where imported into LGD.

3.2.2 MesoWest

MesoWest is a project to provide access to current and archived weather across the U.S. It was build to improve accessibility to meteorologic datasets from North America by unifying and combining data from different station-networks from different organizations and to harmonize the quality control and data format of those stations.[32]
At the moment, MesoWest receives data from 176 different weather station networks, ranging from national networks like the “National Weather Service/Federal Aviation Administration” with more than two thousand stations, over medium size, state-wide networks like the network of the Idaho Transportation Department with 75 stations to small, regional or local networks like the “Salt Lake Community College Mesonet” with just three stations.[86] Also, some non-U.S. networks are connected, such as the Canada Environment network. All networks combined, a total of more than 20.000 reporting stations are available.[87]
Stations connected to MesoWest can report up to 150 different variables.[88] Basic variables reported by nearly all stations are for example temperature, humidity, air pressure or dew point. As different stations and networks are tailored for specific purposes, the sensors installed on stations and so their

[86] An overview of networks can be found at `http://mesowest.utah.edu/cgi-bin/droman/mnet_status_monitor.cgi`
[87] See `http://mesowest.utah.edu/cgi-bin/droman/meso_station.cgi` for a list of all stations
[88] An overview of all variables can be found at `http://mesowest.utah.edu/cgi-bin/droman/variable_select.cgi`

reported variables vary. For example stations that are buoys often have sensors for water temperature, wave period or wave length. Some stations which are near roads, report variables like road temperature or road freezing temperature. Stations also report snow-related variables, like snow dept and snowfall, possibly if they are located in the mountains. If applicable, variables are reported using different scales. For temperature, this is Fahrenheit and Celsius, length are reported both in centimeters and inches and speeds are both reported in miles per hour and knots.
For every station, there is an information page which describes the station. This description includes the spatial coordinates, information about the administrative areas the station resides in, maps of the region the station resides in, images of the station itself, a description of the available sensors including detailed model information, the time the station was installed and calibrated or when was the last time its information was submitted to MesoWest.
For accessing information, MesoWest offers various possibilities. First, current and historic information for every station, can be accessed via a web page or downloaded in XML. Additionally, MesoWest offers the possibility to acquire a data dump for a single station and a certain range of time, from ranges from one day up to one year. These dumps are available as XML, spreadsheet or csv file. The request is submitted via an http-get request, where variables are declared in the request line. Although MesoWest creates and parameterizes the request with a web form, the request can also be created and send from an application, for example a crawler.

4 Enhancement of Gowalla data

This section describes the different approaches we pursued for semantically enhancing Gowalla data with knowledge from LinkedGeoData and MesoWest.
The first approach, which is described in the following section, is to add features to spots, created from information of LGD nodes that match with the spot, i.e. describe the same POI. For example, restaurants in LGD may have an attribute describing its cuisine. By creating a feature from this information and adding it to the corresponding spot, the available information about the spot can be enhanced, possibly helping us to generate more precise models of the user's preferences when using a content-based recommender system. For example, it may turn out that a user often visits restaurants with Japanese cuisine, which would mean that he has a preference for Japanese food. Without the additional feature, this fact would have not been known and could also not be used for content-based recommendation.
For matching spots and their corresponding node, we develop a record linking heuristic, based on the spot's and node's properties. After identifying possible properties that accord, we evaluate different metrics for each property, to identify the one most suitable to us. This is done by using a test set that we created manually. Also using this test set, we create a unifying metrics for the linking task, combining the metrics in an optimal way.

The second approach, described in Section 4.3, creates features characterizing a spot by means of LGD nodes surrounding its position. This features might help to unveil a user's preferences for spots regarding the spot's environment. For example a user might be mainly interested in cafés that lie next to museums and do not like cafés that are near office buildings. This approach has the advantage that for every spot, additional features can be created, regardless if a corresponding node exists for him.

The third approach, adds weather information acquired from MesoWest to check-in information from Gowalla. This might help us to give additional understanding under what conditions user visit what kind of spots. For example, users might tend to visit cinemas and avoid parks if the weather is bad. They also might avoid getting out and stay at home altogether if there is a storm outside. Such context information might therefore enhance the knowledge under what circumstances what recommendation are to be made, predestining it for use in a context-recommender system.
The main tasks in this approach is to identify and acquire the necessary information. For this, we first identify check-ins for which weather information are available, creating a list of weather stations that have spots near them and for each weather station a list of points in time where weather information are necessary. Then, we perform a crawl for obtaining weather data from stations near spots and times a check-in occurred at these spots.

4.1 Linking Gowalla spots to LinkedGeoData nodes

When extracting additional information about a spot from LGD, the first task is to find the node in LGD that corresponds to the spot in Gowalla (i.e. that represents the same entity).
Such a corresponding node might not exist for several reasons. First, it might not have been created yet. As LGD receives its data from OSM, a corresponding node can only exist if a OSM-community member has created it or if it has been part of the data that has been imported into OSM.
A spot might also not have a counterpart because it is not a POI from the perspective of OSM. Spots in Gowalla are created by users, with their individual perspective of which spots should be created and which not. This can include their home (there are numerous spots in Gowalla that are the home of someone) but would not include something like a statue or a bridge, places where Gowalla users will probably not check-in. On the other side, OSM, from which the LGD data origins, aims to create maps from a perspective of general interest. Hence their maps do not include POIs that are only interesting to individuals, but will include those that are interesting to the general public. This also means that such maps will probably include POIs that one would not consider interesting for checking-in in a LSN, like a

statue. In summary, there are spots in Gowalla that will probably not have a corresponding node in LGD but also nodes that will not correspond to a spot in Gowalla. Interesting for us is thus the intersection of both sets, i.e. POIs that are likely to appear in both datasets.

Another issue that might arise with spots and nodes is the whole-part-relationship between objects. A vivid example for this is a shopping mall. Such a mall is composed of multiple stores. Depending on the perspective, the mall can be seen as a single entity (the mall), a set of shops, or a mixture of both. For example, Gowalla might have entries for all or some of these stores, but not for the mall as a whole. LGD on the other side might have an entry only for the mall itself, not for individual stores. Searching for nodes corresponding to spots in LGD would then lead to no result, although a node that "contains" the spots exists. This issue can not be solved easily, as it requires elaborated heuristics to connect POIs with others they are a part of. One would need a good knowledge of the dimensions of the spot to be sure one spot is contained by the other.
But even if this could be solved, features can not be generated from such a match easily. If a set of spots was linked to a node, it is unclear which attributes of the node can be used to enrich the knowledge about the spots. For example if there was a tag "owner" for the mall, adding this knowledge to the spots might be inappropriate, because the owner of the mall is not the owner of the individual shops, but their hirer. In our implementation, we thus ignored the possibility that a POI could be part of another.

Assuming all issues above do not meet for a spot, and there exists a node corresponding to it, the next task is to identify it.
First, it is obvious that the search for a corresponding node will take place in the vicinity of the spot's location. When using a SPARQL, a query with a filter like the one in line 13 in Listing 4 can be used, which returns a list of potential nodes in an area around the spot's position. The approach used then is to compare the attributes of the spot with the ones of the nodes and, based on their similarity, decide which node is the most similar. If the most similar node is similar enough, it can be counted as "matching".
In the following, we examine the attributes of a spot and see if nodes have a corresponding property. For Gowalla spots, Figure 3 in section 3.1 lists the information available. Table 6 in Section 3.2.1 gives an idea over the available information for a node. Although other properties are available, from our experience this example exhibits typical information available for a node, although address information and information about drinks and, and smoking are not encountered frequently.
First, we can sort out all those information that are not in a direct connection to the spot itself but internal data from Gowalla, as they will not help in identifying the corresponding node. Those columns are "spotID", "creatorID", "numberofCheckIns", "numberofPhotos" and "numberOfUniqueVisitors" .Internal keys and flags of the database and data referring to the crawl can be omitted too. Those are "spotFieldsID", "crawlId", "created", "isMerged", "mergedWithSpotID" and "isDeleted".

The remaining fields are "website", "twitterID", "spotCreationDate", "locality", "region", "longitude" and "latitude", "name" and the spot category. The field "website" contains the website of a spot. For a node, there exists a similar predicate called "lgdo:website", which would make it possible to match spots and nodes based on their website. But as only for 2% of all spots a website is denoted, so using it for matchings seems not very promising.
Of all spots, even less than 1% have a twitterID, and there is no corresponding property for a node, which makes the twitterID useless.

For every spot, there exists a "spotCreationDate", i.e. the time when the spot was added to Gowalla. Nodes have a similar property, named "dc:modified". But as the way how spots are added to Gowalla and nodes are added to LGD, it is highly unlikely that corresponding spots and nodes have been added or changed at the same time.

“Longitude” and “Latitude” are the geographical coordinates of both spots and nodes. They are available for every one of them and given the today’s accuracy of GPS, which allows to determine the position of a user within a range of 6m, this information is a first way how we can see if a spot and a node are similar, as one would expect that they are close to each other.
Compared to this, the properties “region” and “locality”, usually referring to the state and the city, are much coarser. Region information are available for nearly half the spots while nearly 90% of all spots have locality information. Given their coarseness, and the fact that geographical coordinates are available, which are much more precise information, these information seem of little use for the matching task.

The name of the spot, its description, the check in radius and the spot category remain. The description can contain additional information of the spot written by users and is available for a quarter of all spots. This is not very much and as there is no equivalent in the nodes of LGD, this information is not very useful. The radius is a value that could give a clue of the size of a POI, as one could guess that the radius is proportional to the size of the spot. Although every spot has a radius, there is no property describing the size of a node, making this also not very useful.
The name of the spot on the other side has an equivalent in the “rdfs:label” property. All nodes also have a label, and only less than 1% of all spots do not have a name. Additionally, the labels of spots are usually personal names, making it easier to distinguish them. The name of spots and the label of nodes are thus the second source that we can use for matching.
Finally there is the category of a spot. 72% of all spots have a category assigned to them and there exist a corresponding attribute, the type property for nodes, which is also a categorization. Although spots have a simlple category and the class of a node can have some superclasses, there is a property called “lgdo:directtype”, which are the types of a node that have no subclasses, which is more comparable to the categories of Gowalla.

Taken together, we have three different pairs of attributes that can be compared. We thus formulate the following hypotheses which will guide us through the creation of the heuristic:

- Matching of spots and nodes based on their names will contribute to the identification of matches between spots and nodes
- Matching the geographical coordinates of spots and nodes will contribute to the identification of matches between spots and nodes
- Matching the category of spots and types of nodes will contribute to the identification of matches between spots and nodes

In the following section, we test those hypotheses. First, we find for every attribute optimal metrics. Then, we evaluate how we combine the optimal metrics from every pair of attributes to generate an optimal unifying metric, performing record linkage based on the three attributes. In order to be able to assess the quality of different matching metrics, we first manually created a test set of matching spots and nodes.

4.1.1 Testdata

For analyzing the hypotheses, a set of test data was generated manually to evaluate the heuristics.
In this process, samples from Gowalla spots were taken from a 25km radius around the center of the city or region of various major cities and metropolitan areas. These areas included London, New York, Austin, Paris, Munich, Berlin, Stockholm, Oslo, San Francisco, Madrid, Rome, the Bay Area near San Francisco, Amsterdam, Dublin, Frankfurt and Leipzig. Those places where chosen if they had a high density of spots and/or a high density of nodes or to add samples from an additional language. Also, we restricted ourselves to cities of countries whose language we could understand at least vaguely, which

often helped in finding the corresponding spots.
The aim was to get a test set of one hundred matches as test basis. In each iteration, at the beginning a sample of 100 spots where taken from some cities and for every spot, a corresponding node was searched in a 500m area around the spot's coordinates. After each iteration, the towns where varied based on the observation of the occurrence of pairs and to bring in other language besides German, English and French. After three iterations, the number of 100 was lowered to 50 for another three iterations. For a sample of the Bay Area near San Francisco, the analysis was aborted because after 35 spots, no match was found.
This strategy led to a total of 435 spots that where examined, finding a total of 104 spots with corresponding nodes from LGD. This means that for 24% of all examined spots, we found a match. It is expected that this quota of matches for the whoel body of spots will be lower for two reasons. First, Europe is with 72% of all samples overrepresented, compared to the 36% of spots they hold. This decision was due to the observation that lgd nodes primarily occour in Europe. Second, the cities analyzed where picked because of the expectation that they have a higher number of LGD nodes. This will not be true for all cities and regions, further dampening the expected quota of spots for that a match can be found.

4.1.2 Matching of Names

The first attributes we will use for matching is the name of a Gowalla spot and the property "rdf:label" of a LGD node. As a spot and a node are independently created in two separate databases, most probably by two distinct persons, one can assume that the naming of the spot and node that refer to the same entity underlies some variation. This may be because of spelling errors or because of variations of the used.[89] Thus, it is necessary to use record linkage techniques to find names that are "similar" to each other. For this task, a wide variety of string similarity metrics have been developed and can be considered. For our purposes, we use the second string library, a java-based package of approximate string-matching techniques.[90] We did not change any implementations, but will point out important features of the implementation if appropriate.
These metrics work by assigning a score between of range $[0, 1]$ to two strings, where 0 means that there is no similarity between the strings and 1 usually means that they are equal.[91]

String similarity metrics can be classified into three categories [21]: edit-based similarity functions, token-based similarity functions and hybrid similarity functions. We will explain and evaluate different metrics from all three categories, explaining their strengths and weaknesses and their performance on the test set we created.

Edit-based metrics

Edit-based metrics work by measuring how many and what kind of "atomic" changes have to be performed in order to convert a string s to another string t.
A well known example of such is the Levenstein metric[38]. It works by simply counting how many insertions, deletions and substitutions of single characters have to be performed to convert string s to string t. For every such operation,-1 is added to the score. The more negative the number is, the less similar two strings are.
As an example, consider a spot with the name "The Cheesecake Factory" and a possible candidate node for matching, which has the label "Cheesecake Factory". Through the addition of the three letters "T", "h" and "e" and the addition of a blank space, the second string can be aligned to the first. Thus, the Levenstein metric has a score of -4 for this pair.
Consider on the other side the two names "UPS" and "CVS", the first being the name of a logistic service

[89] During the creation of the test set it was observed that a frequent variation was that a POI of one dataset had a definite article while the other had not, e.g. "The Cheesecake Factory" vs. "Cheesecake Factory"
[90] `http://secondstring.sourceforge.net/`
[91] It seems that not all metrics assign 1 to equal strings alone, for example Monge-Elkan

provider and the second the name of a pharmacy. The score for those two names is -2, because by replacing the “U” by a “C” and the “P” by a “V” in “UPS”, the two names can be aligned.
These examples illustrate an important drawback of the Levenstein metric, namely that it considers the number of changes independently of the number of characters in the strings. This leads to the believe that the Levenstein metric will not be very suitable for the usage as a name matcher.

To incorporate the length of a string and make the Levenstein metric more comparable, a way to incorporate string lengths in the overall assessment is needed.
A potential solution is as follows: One can examine two strings s and t with their lengths $|s|$ and $|t|$ which are completely different i.e. have no common characters and for which $|s| < |t|$ holds. For aligning string s to string t, one would first change all characters of s to those of t and then add the $|t| - |s|$ remaining characters to s. This represents the maximum number of operations needed to convert one string into another.

Based on this, one can define a scaled version of the Levenstein metric as $1 + d/max(|s|, |t|)$, where d is the Levenstein score of two strings s and t.
For the two examples above, the Scaled Levenstein metric scores:

$$1 + \frac{Levenstein(\text{“UPS”}, \text{”CSV”})}{\max(|\text{”UPS”}|, |\text{”CSV”}|)} = 1 + \frac{-2}{3} = 0.33$$

and

$$1 + \frac{Levenstein(\text{“Cheesecake Factory”}, \text{”The Cheesecake Factory”})}{\max(|\text{”Cheesecake Factory”}|, |\text{”The Cheesecake Factory”}|)} = 1 + \frac{-4}{\max(22, 18)} = 1 + \frac{-4}{22} = 0.81$$

With the scaled version, “UPS” and “CVS” are far less similar, because the number of necessary edits are, relative to the number of characters, numerous. On the other side, the example of the Cheesecake Factory has a relative high score of 0.81, as relative to all characters, only a relative small number of changes has to be made to align those two strings. Because of this improvement, the scaled version is expected to outperform the normal Levenstein metric.
Looking at Figure 6, the expectations are confirmed. The scaled Levenstein metric generally outperforms the standard Levenstein approach, as the Precision-Recall diagram shows.

It should be noted though that the Levenstein metric will fail to assign a high score for some matched name pairs, for example for the strings “The Wheatsheaf” and “Wheatsheaf Hotel”. This is due to the fact that “The “ has to be removed and “ Hotel” has to be entered to the first string to be aligned to the second. This lowers the scaled Levenstein score to 0.375. Opposed to this, if the strings where “The Wheatsheaf” and “Hotel Wheatsheaf”, the score would be 0.75. This example shows that the ordering of words within the string influences the Scaled Levenstein metric, which can become a drawback for string pairs like the example, where, depending on the ordering, the results can differ significantly.

Another edit-based similarity function is the the Needleman-Wunsch metric. It originated from comparison of gene-sequences [25], but can also be applied for name matching [21]. It uses a dynamic programming approach for computing the edit distance of two strings by trying to find their optimal alignment, using previous optimal alignments of the strings’s substrings and their costs.
Given two strings $s = s_1, \ldots, s_K$ and $t = t_1, \ldots, t_L$ where s_i (t_j) is the ith (jth) character of string s (t), a matrix F with $|s| \times |t|$ entries is constructed, the value of $F(i, j)$ being the score of the best alignment between the substrings $s_1 \ldots s_i$ and $t_1 \ldots t_j$.
Finding optimal alignments for substrings begins at the top left corner of the matrix with $F(0, 0) = 0$, optimal alignments for longer substrings are derived from previous optimal solutions of shorter strings,

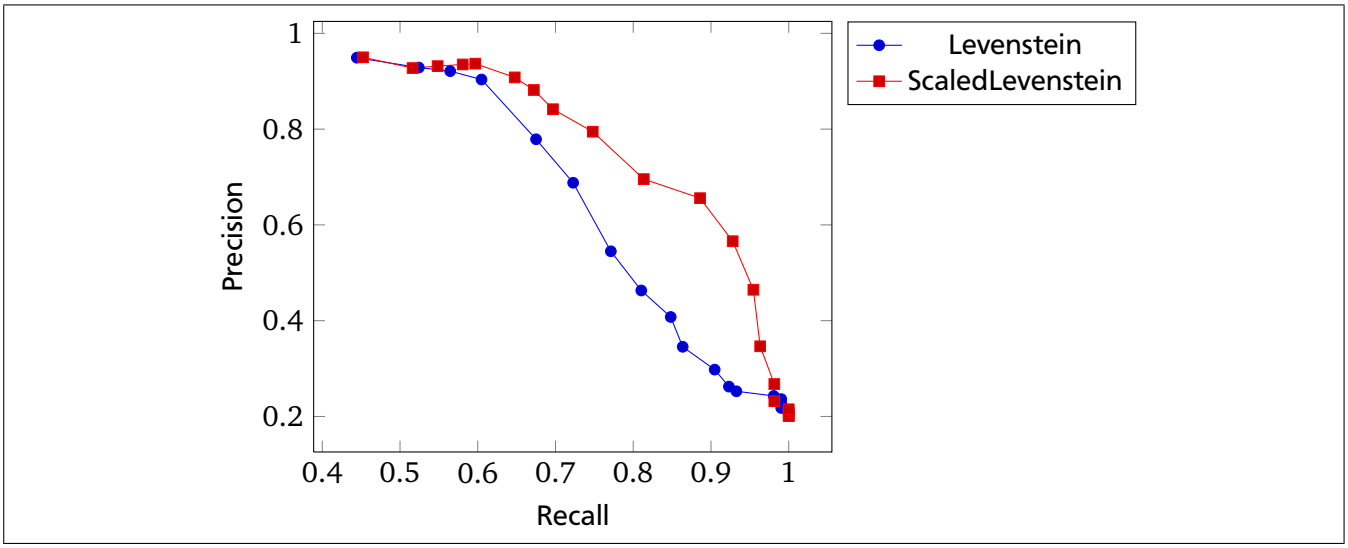


Figure 6: Precision-Recall diagrams of Levenstein and Scaled Levenstein metrics

proceeding in the matrix from top left to bottom right, with the costs of the alignment of the whole strings s and t in the cell $F(|s|,|t|)$.
When computing the alignment score for the position $F(i,j)$ in the matrix, three possibilities to align these substrings exist, as shown in Equation 3.
First, the additional characters s_i and t_j are added to the previous best aligned substrings $s_1 \dots s_{i-1}$ and $t_1 \dots t_{j-1}$. The alignment score is computed by adding alignment costs for the additional characters, $F(i-1,j-1)+s(a_i,b_j)$. The character match score $s(a,b)$ expressed how much it costs to align the two substrings with the characters a_i and b_j and position i and j. The Needleman-Wunsch metric assigns a score of 0 for equal characters and 1 for differing characters.

In the second case s_i is a gap, i.e. it has no corresponding part in t_j. In that case, a gap penalty d is induced, leading to the formula $F(i,j-j)-d$ for this alignment. The third case is the same as the second, only that t_j has no correspondence in s_i. The Needleman-Wunsch metric assigns gap costs of 1. Combined, the equation for computing the alignment score for $F(i,j)$ is:

$$F(i,j) = max \begin{cases} F(i-1,j-1)+s(a_i,b_j), \\ F(i-1,j)-d, \\ F(i,j-1)-d \end{cases} \tag{3}$$

Within the F matrix, there a boundary conditions that have to be treated specially, as for $j=0$, the values $F(i,j-1)$ and $F(i-1,j-1)$ are not defined. As these value represent alignments of a prefix of s to a gap before t, they are defined $F(i,0)=-id$. Likewise we define $F(0,j)=-jd$.
Consider as an example the two name strings “The Fox” and “Fox”. The matrix F can be seen at Table 7. The lower-case numbers near the characters are the boundary values. Beginning at the top left corner, the boundaries score $F(0,0)$, $F(1,0)$ and $F(0,1)$ are considered as initial costs. The marked entires are the optimal alignment costs for the substring from s and t The maximum value is thus 4, which can be seen in the lower right entry. A variation of Needleman-Wunsch is the Smith-Waterman metric, which differs in two ways. [25] First, for the computation of the matrix values $F(i,j)$, an additional possibility is introduced, allowing $F(i,j)$ to take the value of 0, if all other options have a value less than 0:

$$F(i,j) = max \begin{cases} 0 \\ F(i-1,j-1)+s(a_i,b_j), \\ F(i-1,j)-d, \\ F(i,j-1)-d \end{cases} \tag{4}$$

0	T_{-1}	h_{-2}	e_{-3}	-4	F_{-5}	o_{-6}	x_{-7}
F_{-1}	1	2	3	4	4	5	6
o_{-2}	2	2	3	4	5	4	5
x_{-3}	3	3	3	4	5	5	4

Table 7: Example of matrix F for Needleman-Wunsch metric

The second difference lies in the scoring function $s(a, b)$, where matching characters get a score of $+2$ and differing characters a score of -1. This leads to a different results for the example made above, which can be seen at table 8:

	T	h	e		F	o	x
F	0	0	0	0	2	1	0
o	0	0	0	0	1	4	3
x	0	0	0	0	0	3	6

Table 8: Example of F matrix Smith-Waterman metric

A second variation of the Smith-Waterman metric is the Monge-Elkan metric. [6]. For the scoring function, it values -3 for a mismatched characters, $+5$ for matched characters and 3 for approximate matches. Two characters are an approximate match if they are both in one of the following subsets: $\{dt\}\{gj\}\{lr\}\{mn\}\{bpv\}\{aeiou\}\{.,\}$. Also, the Monge-Elkan metric has no constant gap costs d. The start of a gap has a penalty of 5 while the continuation of a gap has a penalty of 1. Lastly, the score is scaled to $[0, 1]$ using the factor $\min(|s|, |t|) \cdot 5$. In our example, the minimal length is 3 which leads to a

	T	h	e		F	o	x
F	0	0	0	0	5	0	0
o	0	0	3	0	0	10	0
x	0	0	0	0	0	0	15

Table 9: Example of F matrix for Monge-Elkan metric

scaling factor of 15. With the results at table 9 the scaled score of the strings is 1.0.

The examples above lead to the notion that Monge-Elkan metric will perform better than the other two metrics for two reasons. First it has the most sophisticated scoring function, which also considers approximate matches for characters that tend to be similar. The second reason is the affine character of the gap penalty, so that longer gaps are,relative to their length less penalized than for other metrics. This makes the approach more robust for if whole words are missing in a name. Although a missing word induces a penalty, the following gap costs are lessened, the length of the string has less impact on the score than constant gap costs would have, as the gap costs do not rise linear with the length of the gap. Regarding the perceived similarity, the length of a missing word is of lesser importance.
Looking at Figure 7, this notion is partly confirmed. Monge-Elkan generally outperforms Smith-Waterman regarding precision, as for every recall and precision value of Smith-Waterman, Monge-Elkan offers a better precision or recall value. Needleman-Wunsch, on the other side generally has offers higher precision values for lower recall values. Although Monge-Elkan generally has good results, from some test matches it produces high scores that are obviously not similar. One of the best examples is the pair "Sheikh Zayed Theatre" and "EAT", which show no resemblance, but for which the Monge-Elkan metric produces a score of 1.0. In general, it seems that Monge-Elkan seems to underrate the differences of strings, which lead to good results for string pairs where one string is a substring of the other. This can

be beneficial if a string is missing, but harmful if this is a coincidence. This also shows in the high recall values up to a threshold of 0.3.

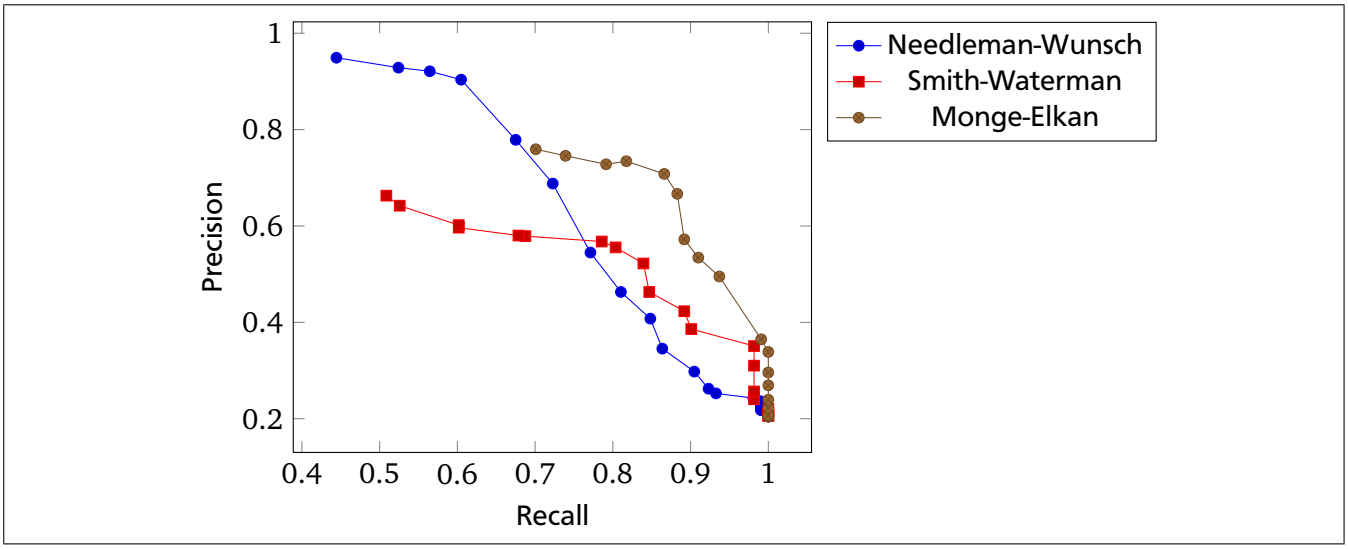


Figure 7: Precision-Recall diagrams of metric Needleman-Wunsch, Smith-Waterman and Monge-Elkan

A metric which is also classified as edit-like is the Jaro metric [34]. It is based on the number and order of common characters between two string. Given the strings s and t with characters $s = s_1, \ldots, s_K$ and $t = t_1, \ldots, t_L$, it defines that a character s_l in s to be common with t if there is a $s_l = t_k$ such that $l - H <= k <= l + H$ with $H = \frac{min(|s|,|t|)}{2}$, meaning that the character s_k occurs in the substring $t_{l-H}, \ldots t_{l+H}$.
Then, let $s' = s'_1 \ldots s'_L$ be the characters that are common with t according to the definition above and let $t' = t'_1 \ldots t'_K$ be defined analog. For every character in s' and t' where $s'_i \neq t'_i$, a transposition, i.e. a switch of the position of two strings, is needed to align the two strings.[92] Let $T_{s',t'}$ be half the number of those transpositions. The Jaro similarity metric for s and t is then defined as:

$$Jaro(s,t) = \frac{1}{3} \cdot \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|} \right) \tag{5}$$

Consider for example the names “Wheatsheaf Hotel” and “The Wheatsheaf”, for which $H = 7$. Figure 8 shows how the characters can be matched, leading to the strings s' = “wheatsheaf hte” and t' = “the wheatsheaf”. As only the second and third character, “h” and “e”, are at the same location, a total of

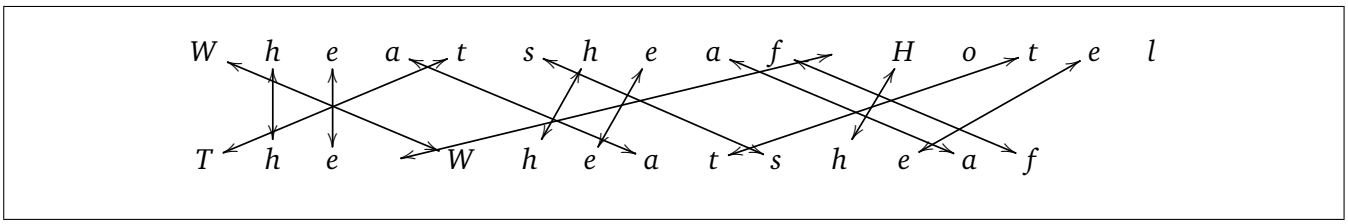


Figure 8: Matching of the strings s and t using Jaoro metric

12 transpositions to align the strings are necessary, so $T_{s,t} = 6$. The Jaro-Similarity is thus $\frac{1}{3}(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{s'-T_{s,t}}{|s'|}) = \frac{1}{3} + (\frac{13}{16} + \frac{13}{14} + \frac{7}{13}) = 0.815$, which is a good value considering every string has a unique word

[92] Remember that in the first step, we created s' and t' in a way that they contain the same characters, albeit in a different order

at different locations.
But as an other example, consider the two strings “The Albert” and “The Black Heart”. Although they seem to be quite different, the Jaro metric assigns them a score of 0.855. This shows that an unfortunate position of characters in two as different perceived words can lead to a high similarity according to the Jaro metric.
A variant of the Jaro metric is the Jaro-Winkler metric [21], defined as:

$$Jaro\text{-}Winkler(s,t) = Jaro(s,t) + \frac{P'}{10} \cdot (1 - Jaro(s,t)) \tag{6}$$

where $P' = max\,(P, 4)$ and P is the length of the longest common prefix of s and t.
With the Jaro metric, strings “Carl von Linne Schule” and “Carl-von-Linné Schule” would have a score of 0.788. Using Jaro-Winkler, one first needs to consider the shared prefix, which is “Carl” , so $P' = 4$. Then, The Jaro-Winkler similarity is $0.793 + 4/10 * 1 - 0.793 = 0.793 + 0.4 * 0.207 = 0.876$. When taking the other example from before, “The Albert” and “The Black Heart”, Jaro-Winkler assigns them a score of 0.913 because they share the prefix “The “.
In summary, the Jaro metrics seem to have some drawbacks, especially because it tends to only look for co-occurrence of characters, although within certain boundaries. The extension of Jaro-Winkler seems to help if the prefix of two similar strings are the same, but if the prefix is something general like “The “, it tends to diminish the differences for the rest of the string. One could guess that this approach will lead to a higher recall, but a generally lower precision, which is even more true for Jaro-Winkler, as the result can only get better compared to Jaro.
In summary, Figure 9 depicts the results for the edit-based metrics discussed above.We omitted Levenstein, as well as Needleman-Wunch and Smith-Waterman as they were inferior. As one can see, scaled Levenstein and Monge-Elkan outperform other approaches regarding the maximum precision. On the other hand, Monge-Elkan is not able to achieve high recall values, where scaled Levenstein is able to have similar high recall values like Jaro or Jaro-Winkler. Those two metrics tend to have the worst precision values when the threshold is tailored for lower recall. Also not that they have a fairly similar precision-recall curve.

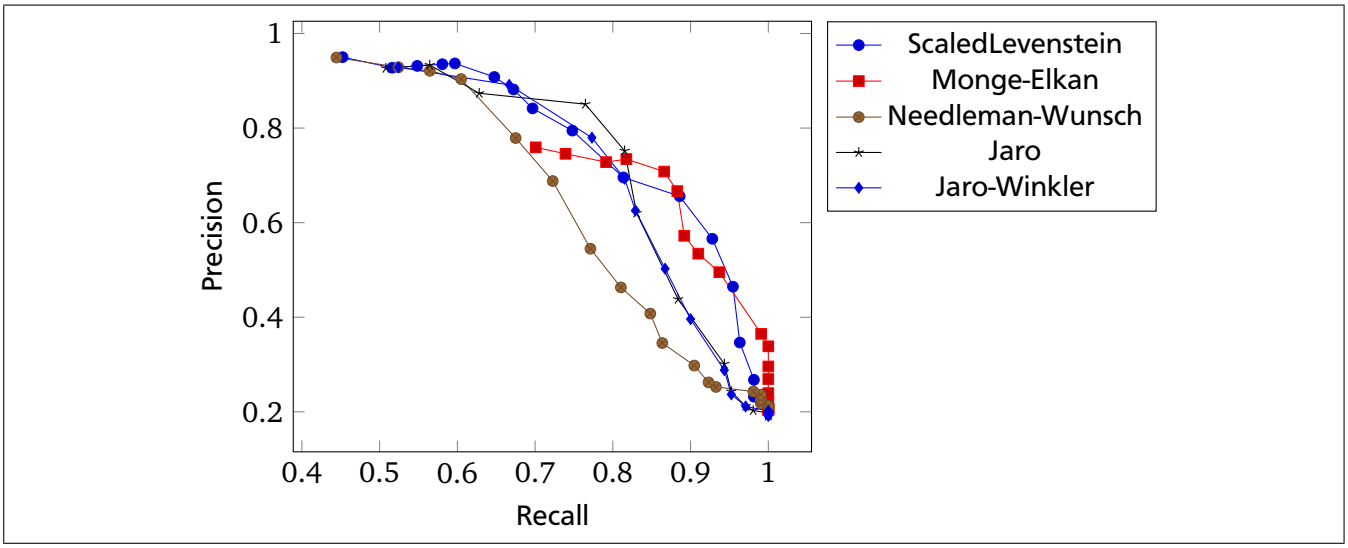


Figure 9: Precision-Recall diagrams of edit-based similarity metrics

Further looking at Table 10, on can see that Needleman-Wunsch performs best regarding average precision, but has the worst maximal F1 value. The Jaro metric on the other side has the best maximal F1 value, but the worst precision. Additionally, Monge-Elkan has both a better maximal F1 and average precision than Jaro-Winkler. In summary, regarding maximum F1, Jaro seems to be the best performing metric.

Method	max. F1	avg. Precision
Scaled Levenstein	0.759	0.447
Monge-Elkan	0.769	0.439
Needleman-Wunsch	0.724	0.453
Jaro	0.793	0.407
Jaro-Winkler	0.765	0.378

Table 10: Comparison of max. F1 and average Precision for edit-like metrics

Token-Based Metrics

Take the example of the hotel “Wheatsheaf”. It may be called “Hotel Wheatsheaf” or “Wheatsheaf Hotel”, which clearly indicates he same hotel. Edit-like distance metrics, taking the name as one fixed sequence of characters, is not able to handle such variances, leading for example to a low score of 0.25 using Scaled Levenstein.
Another way to evaluate the similarity of two names and to avoid the problem of edit-like distance functions is to break them up into substrings, called tokens, and compare the resulting sets of tokens with each other. Such token-based metrics are for example the Jaccard metric, TF-IDF and the Jensen-Shannon similarity.

An important aspect when using such metrics is the tokenizing strategy, defining how the strings are broken up intro substrings. A simple approach is to separate a string into its words, i.e. sequences of alphanumeric characters, which are separated by non-alphanumeric characters, such as symbols or white spaces. Such a tokenizing strategy extracts from the name “Starbucks Coffee” the tokens “Starbucks” and “Coffee” while the name “McDonalds” simply remains “McDonalds”. In the following, such a strategy will be called a simple tokenizer.
Another possibility is to use so called n-grams. They are created by extracting all possible character sequences of length n from a string. For “Starbucks Coffee”, the hexagrams, i.e. n-grams with n=6, would be “Starbu”, ”tarbuc”, ”arbuck”, ”rbucks”, ”bucks ”, ”ucks C”, ”cks Co“, “ks Cof”, “s Coff”, “ Coffe” and ”Coffee”. In the following, this tokenizer strategy will be called n-gram tokenizer. When using such a n-gram tokenizer, finding the right length of tokens is crucial. If *n* is chosen too low, one gets many different, but not very discriminating tokens. On the other side, when *n* is too high, the number of tokens are discriminating but too specialized, making them less useful for comparison between strings.
The tokenizer used in this thesis have some additional features, which are explained in the following.
A simple tokenizer, i.e. a tokenizer which separates words in a string, can have the options to first convert all characters to lowercase or preserve the case. Additionally, it may or may not include single-punctuation characters (i.e. “.” “,”) as stop words.
The n-gram tokenizer works by first applying a simple tokenizer to the whole string. In this context, the simple tokenizer will be called “inner tokenizer”. To each of these initial tokens,a “^” is added at the beginning and a “$” at the end. From each of these tokens, n-grams with a minimum and a maximum size are extracted separately.[93] In our example of “Starbucks Coffee”, first the initial tokens “^Starbucks$” and “^Coffee$” are created. If one uses for example a tokenizer with minimum and maximum token size of six, these tokens are split up into sub-tokens “^Starb”, “Starbu”, “tarbuc”, ”arbuck” and ”rbucks”, which are added to the bag of tokens. The same happens to “^Coffee$”.
If one had chosen a minimum size of four and a maximum size of six, from each token created by the simple tokenizer all tetragrams, pentagrams and hexagrams would be extracted. Additionally, a n-gram tokenizer may or may not add the tokens produced by the inner tokenizer to the resulting bag of tokens. We varied the n-gram sizes in every combination up to a maximum of 10-grams. Although we varied its configuration too, the inner tokenizer had no distinct impact on the performance, and for the following results the inner tokenizer converts all characters in lowercase and the singe-punctuation characters are

[93] The $ at the end is omitted when creating the n-grams

not used as stop words.

The first token-based metric examined is the Jaccard similarity [21]. Given two bags of tokens from two strings, it is computed by taking the number of tokens that appear in both bags and dividing it through the number of tokens that appear in one or both of those bags. With S and T being the sets of tokens from the strings s and t, it can be defined as:

$$Jaccard(s,t) = \frac{|S \cap T|}{|S \cup T|} \tag{7}$$

Consider for example the two strings $s =$ "Harp Bar" and $t =$ "The Harp". The used tokenizer function is a n-gram tokenizer, with a token length of 2 to 4, and an inner tokenizer that ignores the case of characters. The resulting tokens can be seen in Table 11.

Tokens "Harp Bar"	Tokens "The Harp"
"^harp$", "^h", "^ha", "^har","ha", "har", "harp", "ar", "arp", "rp", "^bar$", "^b", "^ba", "^bar", "ba", "bar", "ar"	"^the$", "^t", "^th", "^the", "th", "the", "he", "^harp$", "^h", "^ha", "^har", "ha", "har", "harp", "ar", "arp", "rp"

Table 11: Bag of words for "Harp Bar" and "The Harp"

The Jaccard similarity is then $\frac{|S \cap T|}{|S \cup T|} = \frac{10}{23} = 0.43478$.
The Jaccards similarity returns notably better results than the edit-based metrics for cases where one string is a substring of the other. For example for the string pairs "Place de la Nation" and "Nation", the Jaccard score is 0.468 compared to 0.334 with Scaled Levenstein or 0.4 for Jaro-Winkler. For "Barrylands Railway Station" and "Barrylands", the score is 0.5 compared to 0.38 with Scaled Levenstein, but Jaro-Winkler has a score of 0.87 due to the long similar prefix "Barrylands".
On the other hand, consider the names "Delicato" and "Delikato". The Jaccard score for this example is just 0.4, compared to 0.875 for Scaled Levenstein and 0.95 for Jaro-Winkler. This lies in the fact that a central character is different, making all tokens that have this non-matching character in them not the same in the two sets. Thus, Jaccard is less robust towards spelling errors compared to edit-based metrics.
Another issue is the expressiveness of tokens. Consider the names "Blåbär Café" and "Blåbär". The similarity is 0.62, meaning that a bit more of half of the tokens are shared. This value would drop significantly if the strings would be "Blåbär Restaurant" and "Blåbär", as more tokens from the long word "Restaurant" are just in one of the two strings. But the name "Blåbär" is a very distinct one, which should be considered more important than tokens from words that occur often, such as "Café" or "Restaurant".

To overcome that all tokens are treated equal despite their uniqueness and expressiveness, one can use the TF-IDF-Metric [21]. TF-IDF stands for "Term-Frequency - Inverse Document Frequency", which are the two main concepts for measuring the expressiveness of an token.
Given to bag of words S and T, TF-IDF is defined as:

$$TFIDF(S,T) = \sum_{w \in S \cap T} V(w,S) \cdot V(w,T) \tag{8a}$$

$$\text{with } V(w,S) = \frac{V'(w,S)}{\sqrt{\sum_{w'} V'(w,S)^2}} \tag{8b}$$

$$\text{and } V'(w,S) = log(TF_{w,S} + 1) * log(IDF_w) \tag{8c}$$

The term $V'(w,S)$ denotes the importance of a token. The Term frequency $TF_{w,S}$ is the frequency of a token w in the bag-of-tokens representation S of a string s. The more frequent it occurs, the more

important is the token in characterizing the string. On the other side, IDF_w, meaning Inverse Document Frequency, is the inverse of the fraction of names in the corpus that contain w, where the corpus is the set of all names. The more common the token is in the names of the corpus, the less distinct it is in characterizing the string. Examples for this are the name "Blåbär", which is a very unique token and which probably only appears once, and the word "Café", which probably occurs relatively often. To dampen the effects of both variables, the logarithm of them are used.
$V(w,S)$ denotes the importance of a token w in the string S relative to all other tokens. Multiplying the values for importance of common tokens gives a measure of the similarity of two strings.
When computing the document frequency, it is necessary to build a dictionary of all tokens and their frequency from a corpus, i.e. all names for spots and nodes that are considered for matching. For testing purposes, we used tokens of names of the spots that constituted the test set as well as their matching candidates, i.e. all nodes within 300m of the spot's location.
In the following examples, we assume that the document-frequency has already been computed. Consider the two names "The Capitol" and "Capitol". For this example, we use tokens of length 3 and include the whole words in the sets. the list of common tokens can be found in Table 12. If we wanted

Token	V(w,S)	V(w,T)	$log(TF_{w,s}+1)$	DF_w	$\log(IDF_w)$	$V'(w,T)$
"^ca"	0.176	0.183	0.693	996	2.878	1.995
"cap"	0.370	0.387	0.693	41	6.069	4.206
"api"	0.383	0.400	0.693	33	6.286	4.356
"pit"	0.332	0.346	0.693	77	5.438	3.770
"ito"	0.348	0.363	0.693	59	5.705	3.954
"tol"	0.387	0.404	0.693	31	6.348	4.400
"^capitol$"	0.470	0.491	0.693	8	7.703	5.339

Table 12: TF-IDF Example

for example compute V("tol", "Capitol"), all values for V'(w,T) for all $w \in T$ need to be computed. Those tokens are "^ca", "tol", "ito", "api", "cap", "pit" and "^capitol$". The last four columns in Table 12 show the values necessary for calculating V'(w,T). with these values, $V(\text{"}tol\text{"},\text{"}Capitol\text{"}) = 0.829/\sqrt{1.995^2+4.206^2+4.356^2+3.770^2+3.954^2+4.400^2+5.339^2} = 3.954/10.88 = 0.363$

Like Jaccard, this metric is expected to yield better results for name-pairs where the difference is a missing or an additional word compared to edit-based similarity functions. But compared to Jaccard, it is expected to work better if the the difference of two strings are tokens that are common and relatively meaningless. The previous example, "Blåbär" and "Blåbär Café" shows this very good. The Jaccard similarity is 0.625 because the tokens from the word "Café" where not in both strings and the ratio of tokens from "Blåbär" and "Café" were roughly equal. With TF-IDF, the similarity for these strings is 0.921, because tokens from the word "Blåbär" are considered important, as they are rare, whereas tokens from "Café" are considered less important, because they are common.
But this mechanism has also its pitfalls. Consider the strings "Jollyman Park" and "Jollyman School". The TF-IDF similarity is with 0.792 relatively high, as the tokens of the name "Jollyman" are more unique than "Park" or "School". But it is clearly visible that the names denote something different (a park and a school). The Jaccard similarity for these strings are 0.477. Other examples for this problem are "Ringcenter" and "Kiosk am Ringcenter" (TF-IDF:0.97, Jaccard: 0.58) or "Plaistow Police Station" and "Plaistow" (TF-IDF: 0.683, Jaccard: 0.38).

Another possibility to compare two strings is to see their set of tokens S and T from strings s and t as samples of unknown probability distributions P_S and P_T, which can be compared. Following Dagan et al. [23], the Jensen-Shannon similarity between P_S and P_T can be used as a similarity measure for the

two distributions, which in turn is a similarity measure for the strings s and t.
Let $Q(w) = \frac{1}{2}(P_S(w) + P_T(w))$. Then one can define:

$$Jensen\text{-}Shannon(S,T) = 1 - \frac{1}{2}(KL(P_S\|Q) + KL(P_T\|Q)) \tag{9}$$

where $KL(P\|Q)$ is the Kullback-Lieber divergence, measuring the difference of two probability distributions. For discrete distributions like our case, it is defined as:

$$KL(P\|Q) = -\sum_x p(x)\log q(x) + \sum_x p(x)\log p(x) \tag{10}$$

For estimating the distributions P_S and P_T, different smoothing methods can be applied. We evaluated three methods, an unsmoothed variant using maximum likelihood estimation, Jelinek-Mercer smoothing and Bayesian smoothing using a Dirichlet prior. The first variant simply uses the probability of a token, given a certain word by counting the occurrences of this token divided through the sum of all tokens. Take s = "Fox" and t = "The Fox" as examples. For simplicity, the tokenizer is just a simple word tokenizer.
The probabilities for the tokens are as follows: $P_s(\text{"}The\text{"}) = 0$ and $P_s(\text{"}Fox\text{"}) = 1.0$ for s, $P_t(\text{"}The\text{"}) = 0.5$ and $P_b(\text{"}Fox\text{"}) = 0.5$ for t. Consequently, $Q(\text{"}The\text{"}) = 0.25$ and $Q(\text{"}Fox\text{"}) = 0.75$. The Kullback-Lieber divergences are then $KL(P_s||Q) = 0.415$ and $KL(P_t||Q) = 0.208$ and thus, the similarity for the strings s and t are $1 - (0.5 * (0.415 + 0.208)) = 1 - 0.312 = 0.688$.
Obviously, the unsmoothed version of Jensen-Shannon is generally not robust if the token are just a bit different. For example "Eisenbahnmuseum Kranichstein" and "Eisenbahn Museum Kranichstein" will get a low score of 0.4 because the similarity-distributions just match for the last word.
The other smoothing methods work by discounting the probabilities of a token in a string and then assign the remaining probability mass to the unseen words according to a "fallback" model based on the corpus. Because of this, the fallback model first has to be trained.
The Jelinek-Mercer model for example works by using a linear interpolation of the maximum likelihood model with the collection model, using a coefficient α for each model:

$$p_\alpha(w|d) = (1-\alpha)p_{ml}(w|d) + \alpha p(w|C). \tag{11}$$

where $p_{ml}(w|d)$ is the probability that the token w occurs in the string d and $p(w|C)$ the probability that the token appears in the corpus C used for learning. For our tests, $\alpha = 0.5$ was used, balancing the influence of the maximum likelihood model and the collection model.
Of one assumes that the probability for a token given a word is relative high, wile the probability of a token given a corpus is relative low, this will lessen the difference between the distribution for a given token. Taken the example above, with a learned Jelinek-Mercer model, the probabilities obtained from the corpus might be $p(\text{"}the\text{"}|C) = 0.0233$ and $p(\text{"}Fox\text{"}|C) = 0.0001$. For s this means that $P_s(\text{"}The\text{"}) = 0.5 * 0.5 + 0.5 * 0.02331 = 0.261$ and $P_s(\text{"}Fox\text{"}) = 0.5 * 0.5 + 0.5 * 0.0001 = 0.25005$ and for t, it means $P_t(\text{"}Fox\text{"}) = 1 * 0.5 + 0.5 * 0.0001 = 0.50005$ and the similarity becomes 0.6556.

Alternatively, one can consider Bayesian smoothing using a Dirichlet prior. In this smoothing model, the language model is a multinomial distribution. The conjugate prior for Bayesian analysis is the Dirichlet distribution with parameters $(\mu p(w_1|C), \mu p(w_2|C), \ldots, \mu p(w_n|C))$. The model is then:

$$p_\mu(w|d) = \frac{p(w|d) + \mu p(w|C)}{\sum_w c(w;d) + \mu} \tag{12}$$

For these tests, $\mu = 1$ was chosen. Applying the Dirichlet smoothing to the same example, one gets the same probabilities $p(w|C)$ as $\mu = 1$. But the smoothing approach changes the probabilities the following

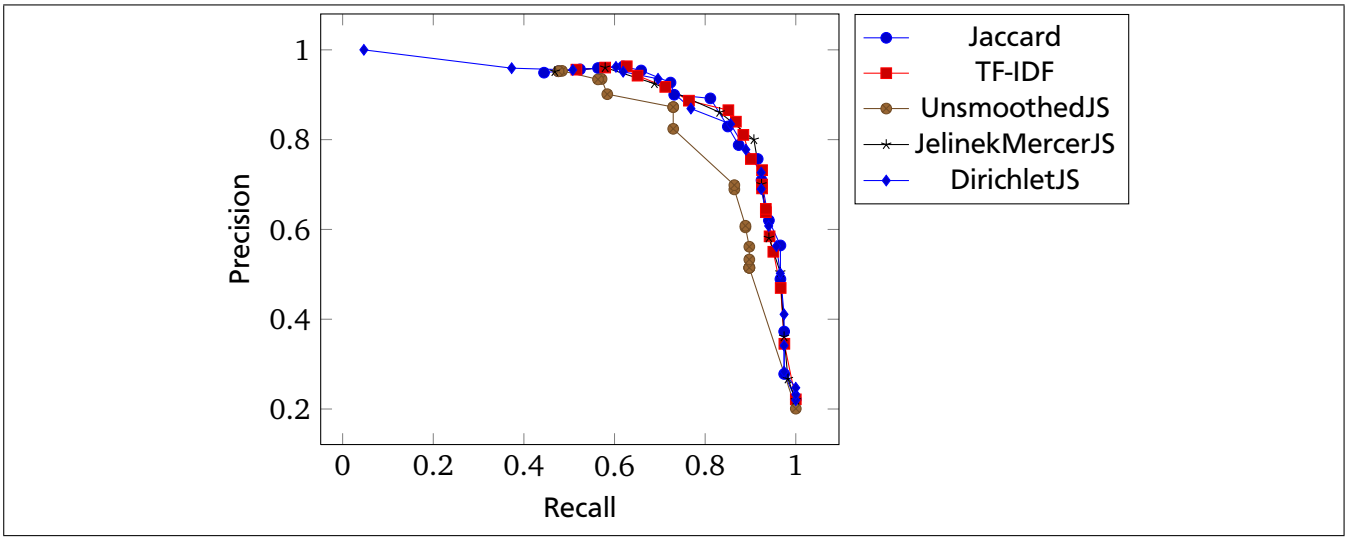


Figure 10: Precision-Recall diagrams of token-based similarity metrics

way: $P_t(\text{"}The\text{"}) = (1+1*0.02331)/(2+1) = 0.3411$ and $P_t(\text{"}Fox\text{"}) = (1+1*0.0001)/(2+1) = 0.333$. This leads to a similarity of 0.595. Figure 10 displays the Precision-Recall diagrams of the discussed token-based metrics.[94]Apart from the unsmoothed version of Jensen-Shannon, the metrics' curves show similar characteristics. As can be seen at Table 13, apart from unsmoothed Jensen-Shannon, their maximal F1 values are fairly similar. For average precision on the other side, Jensen-Shannon with smoothing strategies generally has a low average precision. Also note that the optimal tokens ranges are usually between two and four.

Interestingly, of the best four metrics, Jaccard, having the best average precision, also has the worst maximal F1 value. It can also be noted that Jaccard, begin comparable to the other metrics, is the only metric that is not required to be learned on a corpus.

Also note that token-based metrics outperform edit-based similarity metrics, especially for average precision. The best edit-liked similarity metric as an maximal F1 value of 0.79, where, except from unsmoothed Jensen-Shannon, all token-based metrics have a value greater than 0.82. The best average precision for edit-like similarity metrics was 0.447, where the worst average precision from all token-based metrics is 0.648. This shows that, token-based similarity metrics show better performance on this test set.

Metric	max. F1	avg. Precision	opt. minToken	opt. maxToken
Jaccard	0.826	0.743	2	4
TF-IDF	0.858	0.724	3	3
UnsmoothedJS	0.765	0.708	-	-
JelinekMercerJS	0.850	0.648	2	3
DirichletJS	0.845	0.653	2	2

Table 13: Comparison of F1 and average precision for different token-based metrics

Hybrid Similarity Metrics

The last metric category, hybrid similarity metrics, can be seen as combining edit-like and token-based similarity metrics.They work by first breaking two strings s and t into bags of tokens S and T. Then, similarity is evaluated by applying a secondary similarity function, usually an usually edit-like metric, to the tokens and combining their ratings into one similarity rating. Such hybrid functions are for example

[94] Note that the unsmoothed version of Jensen-Shannon second string only uses a simple tokenizer.

the Level2 function or the Soft TF-IDF metric.

The first hybrid metric used is the Level2 (L2) similarity function proposed by Monge and Elkan [6]. It is defined as:

$$sim(s,t) = \frac{1}{K}\sum_{i=1}^{K} max_{j=1}^{L} sim'(A_i, B_j) \tag{13}$$

where sim' is a secondary similarity function and K and L the number of the tokens from s and t. As secondary function, the edit-based functions introduced above may be used.[95]
In the following example, a simple tokenizer for creating the bag of words and the Jaro metric as a secondary similarity function is used. Table 14 shows the results of the Jaro similarity function for the words of the strings s = "The Wheatsheaf" and t = "Wheatsheaf Hotel". For the Token "the", string s has with 0.622 the greatest similarity. For the token "Wheatsheaf" the identical token is the most similar. According to the formula, these two numbers are added and divided through the number of tokens, i.e. $(0.622 + 1.0)/2 = 1.622/2 = 0.811$. Note that the results vary, depending on the order of the strings used as input. If the order was reversed, the resulting score would be 0.761.
Figure 11 illustrates the Precision-Recall diagram for the Level2 function in conjunction with different

	"the"	"Wheatsheaf"
"Wheatsheaf"	0.622	1
"Hotel"	0	0.522

Table 14: Level2 Example

secondary functions. One can see that the Level2 metric's performance strongly depend on the choice of the inner function. Level2 with Monge-Elkan as inner metric performs worst and is dominated by all other approaches, having both the lowest maximum F1 and average precision. Level2 with Jaro and Jaro-Winkler as inner function have similar precision-recall curves, leading to similar values for maximum F1 and average precision. Finally, Level2 with Scaled Levenstein as inner function outperforms the other approaches in the area of higher precision and generally has both the highest maximum F1 and the highest average precision.
Again, the optimal values for minimal token and maximal token are in the area of two to four.
An alternate hybrid function is the so called "soft" version of TF-IDF. Compared to the "hard" TF-IDF, this

Metric	max. F1	avg. Precision	opt. minToken	opt. maxToken
L2 + Scaled Levenstein	0.878	0.743	2	4
L2 + Jaro	0.858	0.724	2	3
L2 + Jaro-Winkler	0.850	0.708	2	2
L2 + Monge-Elkan	0.765	0.648	2	3

Table 15: Comparison of F1 and average precision for Level2 with different inner metrics

version considers not only tokens that are in $S \cap T$, but also similar tokens, where similarity is defined by a secondary function. Those similar tokens are defined as $CLOSE(\theta, S, T)$, the set of words with $w \in S$ that that have some $v \in T$ such that the the score of a secondary similarity metric, $dist'(w,v) > \theta$. For $w \in CLOSE(\theta, S, T)$, let $D(w,T) = max_{v \in T} dist'(w,v)$, the best value the token w has when comparing it to all tokens in t.
Then the soft TF-IDF-Formula can be defined as:

$$SoftTFIDF(S,T) = \sum_{w \in CLOSE(\theta,S,T)} V(w,S) \cdot V(w,T) \cdot D(w,T) \tag{14}$$

[95] In theory, also token-based metics can be used, but using token-based metrics on small tokens seemed not useful

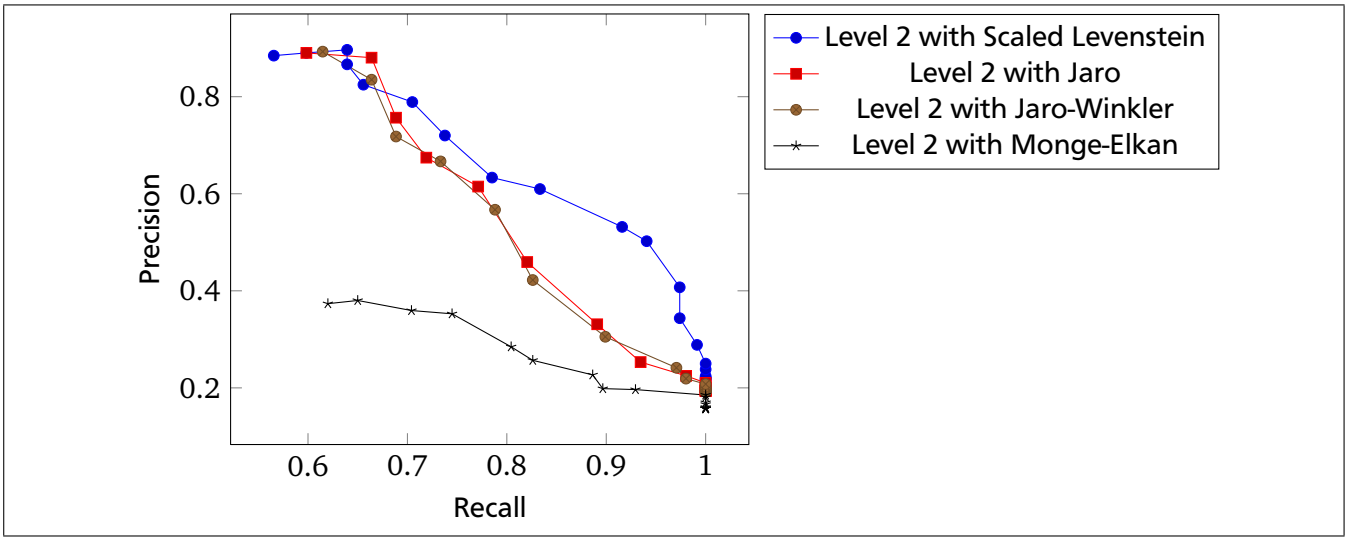


Figure 11: Precision-Recall diagrams of Level2 functions with various secondary functions

Thus within a certain threshold θ, soft TF-IDF also incorporates tokens that are sufficient similar, defined by a secondary similarity function, weighting the score of two tokens by the similarity of each other.

To illustrate this metric, we make use of the following example. Be s = "the capitol" and t = "hte capitol". As a similarity metric, we use Jaro-Winkler and simple tokenizer for creating the bag of words representation of the strings. For θ, we chose a value of 0.85. With the "hard" version of TF-IDF, only the token "Capitol" would be considered, as "the" and "hte" do not match. with $V(\text{"}capitol\text{"}, S) = 0.934$ and $V(\text{"}capitol\text{"}, T) = 0.619$, this would result in a total similarity score of 0.5780.
With Soft TF-IDF, the similarity between the tokens is considered too. The Jaro-Winkler score for the tokens "capitol" in both strings is 1.0 and for the tokens "the" and "hte", the similarity is 0.88. For all other token combinations, the similarity is zero.
Now when using Soft-TFIDF for scoring the two strings, the tokens "capitol" from both strings are matched with a perfect score of 1.0. On the other side, the strings "the" and "hte" are matched with a score of 0.88. Note that in the implementation used, those matchings are fixed, meaning that if "capitol" in the first string is matched with "capitol" in the second string, no other token from the first string can be matched with the token "capitol" in the second string, in order to keep the normalization intact.
As we already pointed out, the two tokens "capitol" have a perfect match with $D(w, T) = 1$. But the tokens "the" and "hte" are also similar with $D(w, T) = 0.88$. Now according to equation 12, this leads to a score of $V(\text{"}\hat{}th\text{"}, S) * V(\text{"}\hat{}ca\text{"}, T) * D(\text{"}\hat{}th\text{"}, T) = 0.164 * 0.183 * 0.6 = 0.018 + V(\text{"}the\text{"}, S) * V(\text{"}tol\text{"}, T) * D(\text{"}the\text{"}, T) = 0.157 * 0.183 * 0.6 = 0.017 = 0.934 * 0.619 * 1.0 + 0.157 * 0.183 * 0.88 = 0.604$.

Figure 12 shows the Precision-Recall diagrams for Soft TF-IDF with various inner functions. Compared to Level2, the different inner functions are more similar in their characteristics. Using Scaled Levenstein as inner function does not show the higher performance like it did when used as inner function for Level2. Taking a look at Table 16, one can see that the maximal F1 values are generally comparable. Also average precision is similar, except for Monge-Elkan, which as a lower average precision that the other three variants. These results are also comparable to Level2 metrics and token-based metrics.
In summary, token-based and hybrid similarity functions seem to be the better choice for linking names of spots and nodes with each other, as they general show high maximal F1 values and a much higher average precision. Edit-based like functions are nonetheless relevant for the use in hybrid heuristics, such as Level 2 or soft TFIDF. Between different token-based and hybrid functions, hybrid distance functions seem to have generally higher maximum F1 values. But as the test set is relative small, this differences do not seem essential. Because of this, we will consider all token-based and hybrid functions in the

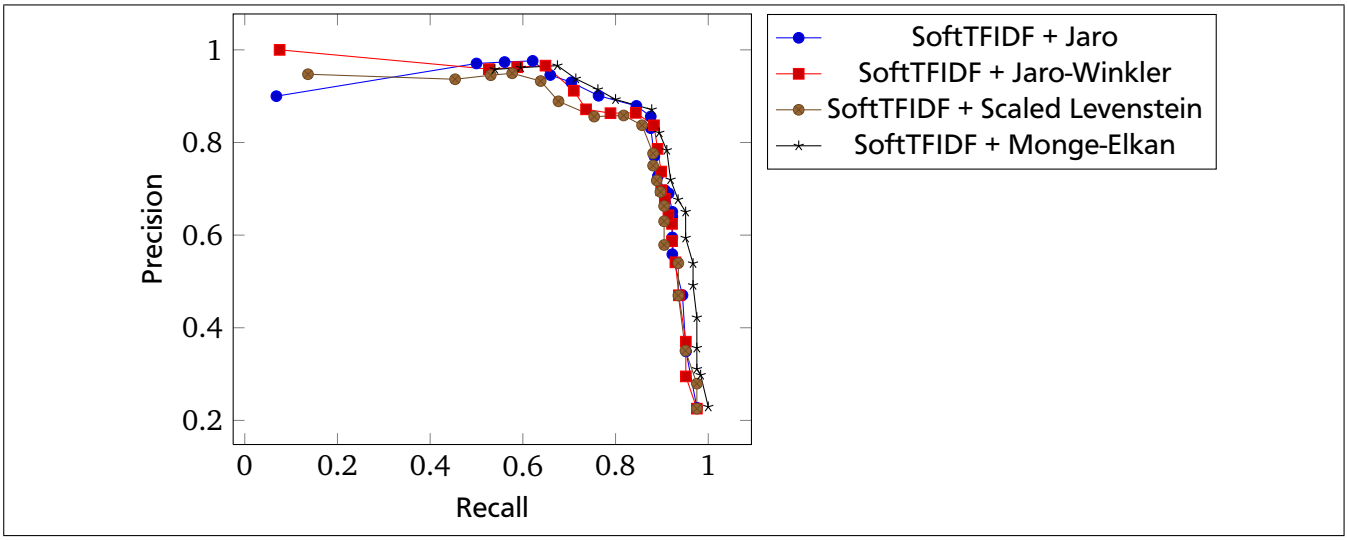


Figure 12: Precision-Recall diagrams of SoftTF-IDF functions with various inner functions

Metric	max. F1	avg. Precision	opt. minToken	opt. maxToken
softTFIDF + Scaled Levenstein	0.847	0.706	1	3
softTFIDF + Jaro	0.865	0.739	3	3
softTFIDF + Jaro-Winkler	0.859	0.710	3	4
softTFIDF + Monge-Elkan	0.8744	0.66	2	3

Table 16: Comparison of F1 and average Precision for Soft TF-IDF with different inner metrics

following unified metric for string similarity. For hybrid metrics, the inner metric which yielded the best results is chosen.

4.1.3 Position of POIs

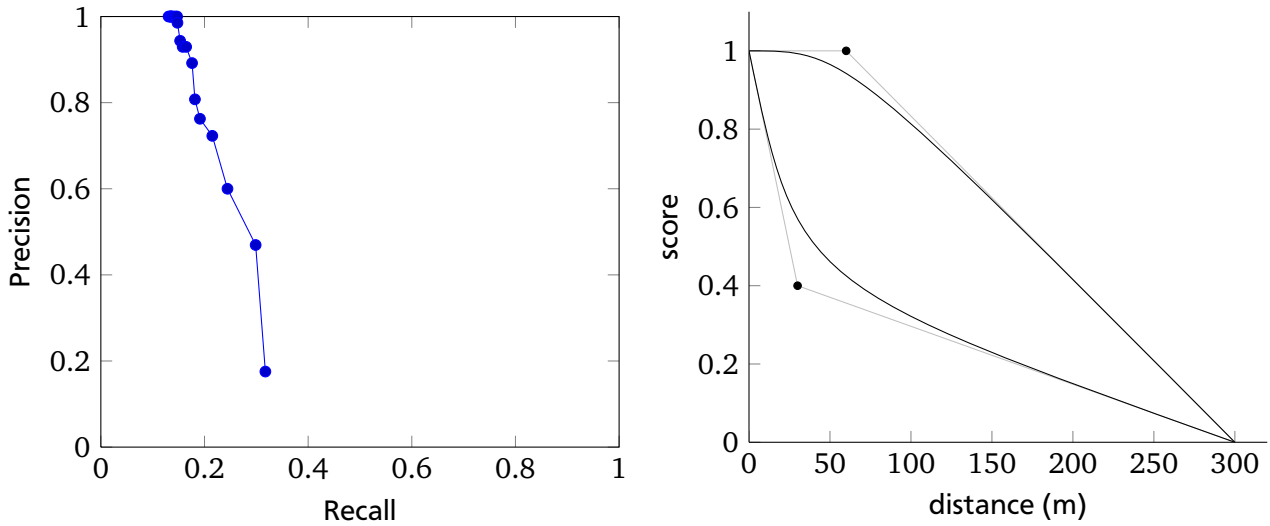
Another way to assess the similarity between a spot and a node is to look at their distance to each other. We compute the distance of two points with the haversine formula, which computes the great circle distance of two points given their latitude and longitude. [1, p. 72]
During the creation of the test data, it showed itself that no node matching a spot occurred outside of a 200m radius around the spot's position. Based on this observation, only nodes within a 300m radius were considered for matching. The additional 100m added to make sure that no potential points were missed.

In order to better compare the nearness of two points with other metrics, the distance between a spot and a candidate node can be scaled to the interval [0.1], where 0 means the distance is maximal while 1 means the distance is zero. We also assume that the nearer a spot is, the higher the nearness score should be. To better control the conversion of distance to a score, it may be beneficial to allow the scoring function to be not linear and to learn the optimal curve through our test data. For example it may be optimal to assign spots that are in a 10m radius around a spot a similar high probability due to measurement errors, while the score difference between nodes with a distance of 100 and 110 meters should be higher
For this, we implemented a Bézier curve. A Bézier curve of degree n has n+1 so called control points. It describes a polynomial of degree n, which runs through the control points 1 and n, and whose shape

can be controlled by the remaining control points.
Bézier-Curves are generally defined as:

$$C(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i, \quad with \quad B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{15}$$

For our purpose, we use a quadric Bézier curve (n=2), which creates a parabola, running through the first and last control points. We define these points as $P_0 = (0,1)$ and $P_2 = (0.3,0)$[96], which states that for a distance of zero, the score is 1 and for a distance of 300 meters, the score is zero. With the remaining point, we can no control the shape of the Bézier Curve. Figure 13b gives examples of two Bézier Curves along with their control points. As one can see, the free control point drags the curve in its direction, allowing to easily control the shape and thus the conversion of distance to score. For the concrete implementation of the quadratic Bézier curve, one can refer to the appendix.
Two things should be noted. First, the values of the Bézier-Curve will take only valid values, i.e. it will not be possible that a distance value will get assigned a value that is outside the $[0,1]$ interval. This is because the curve is within the convex hull that the two points P_0 and P_2.
Second, as the relation $x_0 < x_1 < x_1$ holds, where x_i is the x-value of the control point P_i, the curve is monotonic.
Using this cure, we are able to systematically variate the scaling of the distance. For distance alone, an optimal curve was found with the control point at (0.06, 0.4). The lower curve in Figure 13b shows the corresponding Bézier-Curve. Figure 13a shows the Precision/Recall diagram for the distance metric. The maximal F1 value is 0.3650 with a average precision of 0.17.
Compared to string metrics, both maximal F value and precision is considerable lower. This may be due to the fact that the ordering of possible matches is determined, regardless of the shape of the scoring function. Nonetheless, considering the distance as part of a unified metric may serve for getting better results than using a string metric alone.



(a) Precision/Recall diagram of distance metric

(b) Example for different Bézier Curves

Figure 13: Precision-Recall diagram for distance metric and example Bézier Curve

4.1.4 Categories of Spots and Nodes

A third way of assessing the similarity of a spot and a node is to compare the categorical information of both.

[96] We measured the distance in kilometer

As already described in Section 3, the categorical information of the two systems differ structurally. In Gowalla, a spot can have one or no category assigned. There exists 457 different categories, which have no typological order[97]. On the other side, a node from LGD can have one or more types, according to an ontology, and they can have more than one category. A restaurant that has a bar attached, is expected to be of type "Bar" and "Restaurant". These classes, in turn, are subclasses of "Amenity", which in turn is a subclass of "Node".

Although the structures of the categorical information are different, they show a high degree of similarity for the same entities. For a spot with category restaurant, the corresponding node will often be of type restaurant. Taking all spots of type restaurant with a corresponding node, most nodes will probably be of type restaurant. By exploiting these correlations, the categories become comparable to each other. For this purpose, we used association rules. [5]
Association rules origin from market basket analysis of customer transactions. Such transactions consist of sets of items that customer have bought together. Based on such past data, on can mine association rules like "90% of transactions that contain bread and butter also contain milk".
Association rules are defined as follows. Let $I = I_1, I_2, \ldots, I_K$ be a set of binary attributes and let T be a database of transactions. Each transaction t is a binary vector, where $t[k] = 1$ if t incorporates the item I_k and $t[k] = 0$ otherwise. Further, let X be the set of some items I. Then, the transaction t satisfies X if for all items I_k in X, $t[k] = 1$. An association rule is an implication of form $X \Rightarrow I_j$ where X is a set of items in I and I_j is a single item in I that is not present in X. The rule $X \Rightarrow I_j$ is satisfied in the set of transactions T with the confidence factor $0 \leq c \leq 1$ if at least c percent of transactions in T that satisfy X also satisfy I_j.

For these rules, syntactic and support constraints can be applied. Syntactic constraints mean that only certain items allowed to appear in the antecedent (X) or in the consequent(I_j). Thus, for a rule $X \Rightarrow I_j$ one can specify $I_{antecedent} \subseteq I$ with $X \subseteq I_{antecedent}$. Additionally, we specify $I_{consequent} \subseteq I$ with $I_j \in I_{consequent}$.
Support constraints mean to impose restrictions on the minimal or maximum number of transactions from T that support a rule. After the creation of such rules, all those are rejected that do not satisfy minimum values for c, e.g. $c_{min} \leq c \leq c_{max}$.

One can adapt this idea to match spots and their categories with nodes and their types. In our case, the transactions T would be the set of all matches t that could be discovered. I consists of all categories from Gowalla and classes from LGD. For a transaction t, $I[k] = 1$ if I_k is either a category of the spot s or type of the node n.

An example for such a transaction would be $t =$ ("Bar", "http://linkedgeodata.org/page/ontology/Bar"). Assuming that in 90% of all matches where a spot has the category "Bar" the corresponding node also of type bar. Then, this forms a rule with a confidence factor of 90% or alternatively formulated: "90% of transactions that have the gowalla category bar as antecedent have the LGD class http://linkedgeodata.org/page/ontology/Bar as consequent". As we have defined appropriate syntactic constraints, this can be translated to "for a Gowalla spot with the category "bar", the probability that the corresponding node has the category "http://linkedgeodata.org/page/ontology/Bar" is 90%".

One problem is that with the existing test data, usable association rules can hardly be obtained due to the size of the training set. With only 104 matches, compared to 457 Gowalla categories and even more categories from LGD (apart from the possibility that a LGD-Node can have more than one category), more data is needed. As it is not feasible to acquire the necessary data manually, an alternative

[97] For movies, activities and interactive events, there exists categories like "Film > Screening > Documentary Short", but they apply only for 1157 or 0.00006% of all spots

way to automatically get a large number of spots and matching nodes has to be found. Such matches should have a very high precision, making the recall only of secondary importance, although an adequate number of matches should be obtained. In order to meet these requirements, a special record linkage heuristic was used. This heuristic only considers nodes within a 50m radius and matches them with the spot if it is the only one that has an equal name. Running this heuristic, we obtained a total of 59617 matches, which are 3% of all available spots.
On these set of matches, the a priori program from Christian Borgelt was used[98],configuring that only spot categories are allowed as antecedent and only node categories as consequence. No restrictions where imposed on the minimum number of supporting transactions for a single rule. With this configuration, a total of 4145 rules where created.

Examples for the results are for example that the category “Traditional Art Museum” from Gowalla co-occurs with the type “TourismMuseum” in 82.2% percent. The category “South American/Latin”, which seems to refer to restaurants, co occur in 4.9% with the type “Cafe”, in 6.9% with the type “pub” in 3.9% with the type “FastFood” and 84.3% with the type “Restaurant”.
On the other side, you have for example the type “building”, which co-occurs with many different Gowalla classes, including “Steakhouse” (0.3%), “Bank and Financial” (0.2%) , “Mall” (0.9%) and others.
Using these rules, the category metric alone has a maximal F1 value of 0.365 and an average precision of 0.17. These results are similar to the results for the distance metric, but again this does not mean that the category metric will not contribute to an unified matching metric.

4.1.5 Unifying metric

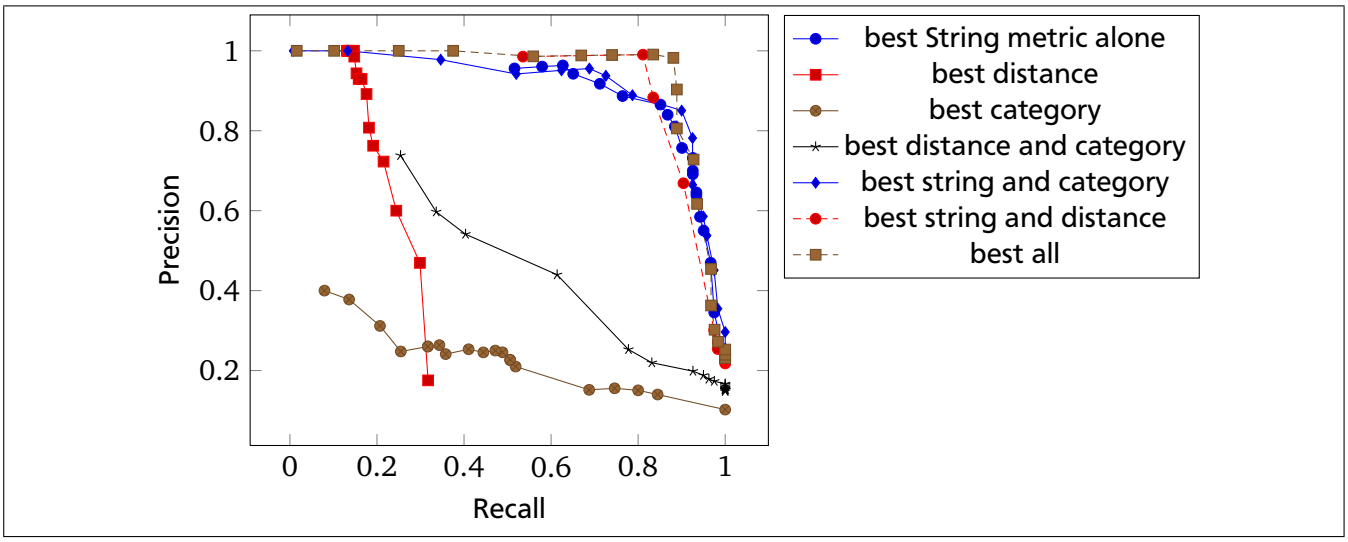
Finally, the above constructed metrics are combined into one unifying metric. The different metrics were combined in a linear framework, where the weights of metrics sums to one.
The following parameters where available:

- **String metric**: Every string metric that showed good performance was considered. This includes Jaccard, Soft TF-IDF, DirichletJS, JelinekMercerJS, Level2 function with Scaled Levenstein as inner function and SoftTFIDF with all inner functionst’mentioned above. For those metrics, the configuration which yielded the best results was used.
- **Distance Metric**: The position of the second control point. Every value for the x between 0 and 0.3 in steps of 0.01 and y coordinate between 0 and 1 with steps of 0.05
- **Metric Weights** The metrics where weighted individually, with the score of all weights summing up to 1.0 For testing, the weights where varied with steps of 0.1.

With this set-up, the optimal combination of metrics and weights where searched. Figure 14 shows the Precision/Recall diagram for the different runs, testing if omitting one metric harms the overall result.As best string metric, we used TF-IDF, as it yielded the best results in the unifying metric.
A first observation is that the distance metric and category metric alone have the worst results. As can be seen in Table 17, they have the worst max. F1 and the worst average precision. Taking a look at Figure 14, one sees that distance as a nearly constant albeit not high precision, while recall is, depending on the chosen threshold, between 0.2 and 1. The category metric on the other side is able to produce higher precisions, while not having a recall higher than 0.4. These results are probably explained by the fact that spots of similar categories have a tendency to cluster. For example in a shopping area, there will be multiple shops, which all have the same category. In such a case, identifying a match based only on the category will not be very efficient.
The string metric on the other side seems to have good results even alone, exceeding the other two

[98] www.borgelt.net/apriori.html

approaches. This shows that the name is of great importance when identifying entities. Intuitively, one would probably not put for example a shop next to another and name them alike, and as the spots' and nodes' names are often personal names, the naming seems to be quite consistent.
Next, we look at the combination of two metrics. As one can see, using only distance and category boosts maximal F1 and also improves average precision. The combination of both seems to compensate each others shortcomings, as can be seen from Figure 14, where their combination is dominating both approaches. The optimal weights of the two metrics show that distance is more important, having a weight of 0.8%.
Adding distance or category metric to the string metric does also improve performance, but not as strong as for distance and category. When adding on of them , distance has a more positive impact and has also a greater weight than the category metric.
Finally, the unifying metric has the best results regarding maximum F1, but sometimes seems to be inferior to an only string approach. Interestingly, the weights are distributed in a way that the string metric has a weight of 0.5, while distance has an influence with 0.4 nearly as big, while the category has only a weight of 0.1. This shows that especially the name and the distance are of great importance.
Taking an overall look at the combination of distance metrics, one result is that only using two metrics is generally inferior to using all three. Our hypotheses that every metric contributes something to the matching of spots and nodes are thus confirmed.
Using the optimal metric outlined above, for every spot in Gowalla a corresponding node was searched.



Metric	max. F1	avg. Precision	str.metric	point	Weights		
					string	distance	category
distance	0.365	0.17	-	-	0	1	0
category	0.327	0.236	-	-	0	0	1
string	0.878	0.743	L2+S.L.	-	1	0	0
dist. and cat.	0.512	0.254	-	(0.06,0.1)	0	0.8	0.2
string and cat.	0874	0.680	JSDirichlet	-	0.9	0	0.1
string and dist.	0.89	0.36	TFIDF	(0.18,0.1)	0.8	0.2	0
all	0.929	0.7	TFIDF	(0.03,0.4)	0.5	0.4	0.1

Table 17: Comparison for unified metric while including different metrics

This resulted in finding a total of 176468, which are approximately 10.24% of all spots. These spot

accumulate 2.6 million checkins, which are 16% of all check-ins. With all spots have an average of 9.6 check-ins, spots with a corresponding node have an average of 15 check-ins, which indicates that nodes tend to be found for spots which are popular above average. Figure 27 in the appendix displays the heatmap of the spots where matches where found. Interesting the highest density of matches are in London. Also Germany seems to have high densities of matches. The U.S. has matches in bigger cities, but there do not seem to me as many matches as in Europe.
This result does not come as a big surprise. If one would overlay the heatmaps of spots and nodes, areas with high densities are both London and Germany. On the other side, Stockholm, which showed a high density of spots in Gowalla, does not have a high density of nodes from LinkedGeoData, resulting in a lower density compared to London. Apart from bigger towns in the U.S. and central Europe, there matches are not found frequently in other parts of the world.

4.2 Feature Generation

After linking spots and nodes, we can now generate additional features for spots based on nodes matched. For performing this task automatically, FeGeLOD, a tool generated by the Knowledge Engineering Group at Darmstadt University of Technology, can be used. [49]
FeGeLOD , short for "Feature Generation from Linked Open Data" is a toolkit for unsupervised generation of data mining features from Linked Open Data. It is implemented as a filter for Weka, a data mining tool.[99] Initially, it was tailored to automatically generate features from entities in DBPedia for a list of database columns. We modified it so it could be used with LinkedGeoData.
Feature generation in FeGeLOD is compromised of three subsequent steps. In the first step, entity recognition is performed in order to find corresponding entities. In the second step, features are generated, based on the identified entity. The last step is a selection step, where a subset of generated features is selected based on their quality.

The fist step, entity recognition, initially was performed by taking a feature from the initial dataset and use a heuristic to link it to an appropriate entity in DBPedia. As we already linked spots to nodes in the section before, we skipped this step.

The next step consists of generating features based on the linked entity. For this, FeGeLOD offers six different feature generation strategies. Two of them are entity-based and work directly on the attributes of the identified entity, while the four others are relation-based and take other entities, connected to the identified entity, into account.
The first entity-based generator creates features from the properties of an entity. In the initial generator, string-features are ignored, as free-text usually cannot be used as features. We omitted this limitation, as the information that such properties hold, if necessary, can be transformed into a bag-of-words representation.
The second entity-based generator creates features from the types of an entity. For a node from LGD of type "lgdo:restaurant", the features "lgdo:restaurant", "lgdo:amenity" and "lgdo:Node" as the latter ones are super-classes of restaurant (and "lgdo:amenity" is a subtype of "lgdo:Node").

The other four strategies take relations of entities into account. The first two generators consider relations in which an entity is a subject or an object. One generator creates a binary feature which indicates if the relation exists, while the other creates a numeric feature, describing how many relations exist from or to other entities.
The last two generators deal not only with the connected entities itself, but with their types. If a node is related to another entity, for example a country, the type of this entity, i.e. "country", would be used as a feature. Like the two strategies before, one strategy generates a binary attribute while the other

[99] http://www.cs.waikato.ac.nz/ml/weka/

generates a numeric attribute.
In a last step, a subset of all generated features is selected. This is necessary as the generators may create a large number of features, not all being useful for adding relevant knowledge about a spot. For example, a feature that as only one value for each spot will not be very useful, while on the other side, every node will be of type “lgdo:Node” which will also not add much knowledge.
For this task, a basic heuristic is used. For a given threshold p, all features will be discarded that have a higher ration for missing, identical or unique values than p.

For application with LGD, one can formulate some expectations on how the different generators will perform. The first generator, using properties from features, will probably add the most interesting information. This concerns especially properties such as the type of cuisine of a restaurant or the accessibility of a node. The features of the second generator, in turn, will be closely coupled to the category field that a spot from Gowalla has. Here, the possibility to “refine” the data from Gowalla, for example by adding that a certain restaurant is also of type bar, will add information.
The other four generators, are not expected to generate many features. This is due the fact that entities in LGD are not strongly interlinked with each other. The only links observed where the “contributor” relation, which shows which contributor added or modified a node and the “lgdo:wheelchair” relation, which classifies a node regarding his wheelchair accessibility.

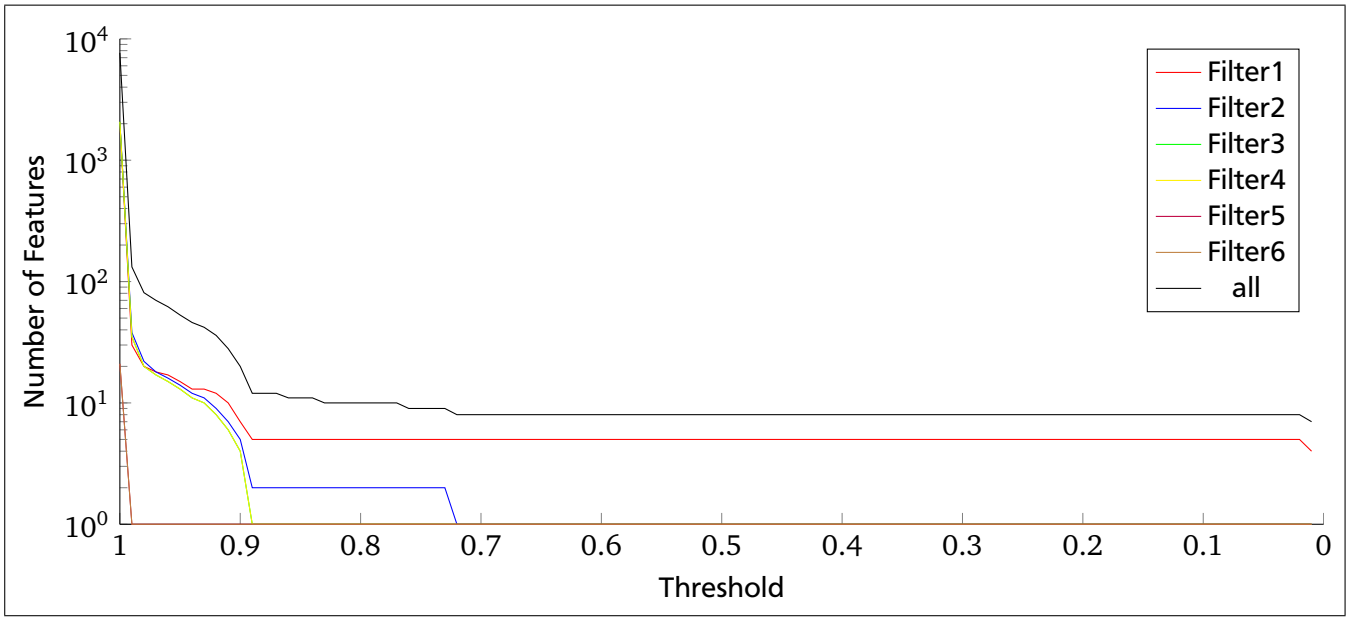


Figure 15: Number of features generated by different filters depending on the threshold

Figure 15 shows the results of the generated features depending on the threshold chosen for the filter. As the scale is logarithmic, the lowest value is 1, which is the case if just the nodeID (the input) remained and no other feautres were created.
The first observation is that the number of features is declining rapidly if the threshold lowered from 1.0 (which would allow all features) to a value of around 0.99. This is not surprising, as many features seem to appear exactly once in all nodes. It can also be seen that features generated from strategy five and six are rejected instantly, dropping from 21 at a threshold of 1.0 to 1 at a threshold of 0.99. These features described the type of linked entities. It turns of these types are usually XMLNS definitions for literals and are 99% the same. Thus features from those two generation strategies are not usable for our purposes.
Strategy three and four deal with relation to other entities. Here, compared to features from strategies five and six, for lower thresholds, some features are not rejected instantly. The best features on a 0.95 level are those that state if there is a relation to some address related entities, such as a postal code, a

city, street or a house number. Also, features that state if the place has a website and a cuisine description are are features that have a lower metric score than 0.95. On a level of 0.9, only features indicating the number of properties linking to street or house numbers and the link to GNIS-IDs remains. GNIS stands for Geographic Names Information System and are entities that nodes can be linked to, indicating geographical and cultural geographic features in the US. Such features include the state, county, and the street. The fact that for threshold values smaller than 0.95 the binary strategy produces less features than the numeric strategy is due to the fact that the number of types has a higher variance than the question if a type is available or not.
Strategy two created features of the types of entities. On a level of 0.9, only the types amenity, "shop" an "restaurant" remain as features. for threshold of 0.95, more features are not rejected, for example if the node is of type "FastFood", "Cafe" and "Pub", "RailwayStation", "Landuse" or "Railway".
Lastly, strategy one looked at the values of a node's attributes. Here, on a 0.9 level, many features resemble ones that are already included in the spots information, such as "label", "latitude" and "longitude". The two exceptions are the feature which describes the street, which has 89% missing values and the feature ID fro GIS, which has similar value. For a higher threshold of 0.95, the node's address details are included as features, as well as its website and additional GNIS features, such as state, county.
In Figure 15, the number of features from all strategies combined can be seen. For a threshold of 0.95, a total of 46 features remain. From these features, a third are node information (strategy one) twenty percent are type descriptions (strategy two), both a quarter come from strategies three and four and 16.% come both from strategy five and six. For a threshold of 0.9, sixteen features remain. Six are from strategy one and are "label", "geometry", "latitude", "longitude", the node's street and the GNIS feature ID. Three features are from strategy two, indicating if the node is of type amenity", "shop" or "restaurant". For strategy three, four features remain, indicating the number of types a node has, the number of links to street entities, house numbers and GNIS features. For strategy four, the the same binarized features are created, except for a binary feature indicating if a node has a type, which is always one. Strategies five and six do not create features with a lower metric score than 0.9

In the end, we chose to import all features that have a metric value higher than 0.999. The reasons why such a high value was chosen are the following. First, lower threshold resulted in only a few features, and those features tend to have a corresponding feature in the Gowalla dataset. So if using such low threshold, barely any new knowledge could be created. Second, having features that are only shared by a very low number of nodes does not need to be a disadvantage. This is because the spatial distribution of features is of importance. As it has been noted in the literature[100], many users of LSNs tend to have areas of higher activity and when recommending items to a user, compared to for example a movie rental website, not all available spots need to be considered. Consider a case where all spots with a feature describing a restaurant's cuisine are found in one city. Although, compared all spots, a very low fraction of spots have such information, users in this town can profit from such a feature, while persons in other areas will not. Hypothetically, if on the other side this feature is evenly distributed over the globe, it will probably not help to describe a user's preferences, as the probability that a user will visit more than one node with such a feature will be considerably lower.

4.3 Creating an environment from surrounding LGD nodes

The second approach for creating additional features for spots from Gowalla is to use the nodes' types in the vicinity of a spot for describing the spot's environment. It is hoped that the environment of a spot, seen through the perspective of the LGD nodes located there, will be discriminating and will unveil user's preferences or distastes for spots.
For example for spots that are located in shopping centers, we expect to find nodes of type "shop" in the vicinity. For restaurants that are located at Main Street, one would expect to see a mixture of restaurants,

[100] See Section 2.4.1

bars and shops. A restaurant that is frequently visited by tourist as another example would probably be located in the vicinity of tourist attractions.
Based on the number and structure of nodes in the vicinity, the users preferences might be visible. Users may be attracted by the surrounding or might try to avoid spots that lie in a certain environment. For example a user might like to go to a restaurant in an amusement center as he might want to visit a bar after eating dinner at a restaurant on a Friday noon. On the other side, a person living in Rome might want to avoid restaurants that are in an area with many tourist attractions because of their high prices.

To exploit the additional information from LGD, for every spot the nodes within an 100m radius were obtained. From these nodes, the frequency of the direct types were obtained, giving an impression of the spot's environment. This method leads to an average of around 5 types in the area of an spots.

Take as example the Apple Store in Coven Garden, London. Taking a look at Gowalla spots in the environment, you find a lot of uncategorized places which turn out to be shops. Using LGD on the other side, it shows that the direct vicinity houses boutiques, fast food restaurants, shops for jewelery, shoe restaurants and hairdresser.
An or take the city palais of Darmstadt. In Gowalla, there are just the ULB in the vicinity. Using LGD nodes, you find nodes of category attraction, castle, concert hall, library, pub tourism museum and two tram stops.

4.4 MesoWest Data

In contrast to the data from LinkedGeoData, which we used to enhance our knowledge of a spot, data from MesoWest can be used to enhance the available information about check-ins in Gowalla, enhancing our knowledge about the context of a check-in. The aim is thus that for every check-in, we will try to find the prevailing weather situation.
For this task, some things need to be considered. First, MesoWest is a network of weather stations primarily located in the U.S. and Canada. Thus it can be expected that we will find information about check-ins only from spots that are in North America.
Another factor is the heterogeneity and availability of data. As MesoWest bundles more than 150 networks of meteorological stations and both networks and their stations have different characteristics regarding available variables and available observation times. The first point is not a problem, as there seems to be a core set of sensors that are installed on every station, usually temperature and humidity-related.
The second point is more relevant. Although the earliest available data is from 2002, the time when a station was first reporting data to MesoWest varies, both because of the time when the network was added to MesoWest and because of the time the station was added to the network. So it can happen that a station which is near a spot did not report data when some or all check-ins occurred.
Based on these observations, we advanced to link stations to spots as follows.
First, in order to decrease the number of necessary requests, from all 175 available network, only those with more than 500 stations were considered. This resulted in using stations from seven networks with approximately 15,000 stations or around three quarters of all available stations. Without this constraint, two spots that are in the same area may be linked to different stations and if they have a check-in performed at the same time, two request for data are necessary. With this constraint, we can reduce the number of requests, lowering the load for the crawling process.
Second, we excluded all stations that where not set up before the first recorded check-in, which was performed on 24.1.2009. Although it may be possible that a station was set-up after this time, but before the first check-in at a spot near to it was performed, it reduced the number of necessary requests further and also eased the data acquiring process, as the time of set-up of stations did not need to be considered for every station individually.

Having a list of stations than can be linked to their nearest spots, we proceeded s follows. The first step for crawling weather data was pairing every spot with its nearest station. As a threshold, a distance of 30km was the maximum distance allowed, otherwise a spot was considered to have no station nearby. This resulted in being able to find a station for 917,200 spots or 45 percent of all spots. This number is closely resembling the number of spots in the United States, which is 48.41% of all availalbe spots. The average distance between a spot and its station is 11,5km, which seem to be near enough for the weather predominating at the station's location being sufficient similar to the weather at the spot's location.

After this, for every station, the day, month and year of every check-in from every spot that was paired with the station was computed. As the used API could return data for an 24-hour period, the weather observation where obtained day-wise. This generated around one million requests, which consisted of a stationID, a day, a month and a year. The request returned a XML containing a list of observations with a list of measurements taken from the available sensors. The request was of a form that all information available was crawled, so no prior knowledge of available sensor data was necessary.

This data was crawled, using a web mining farm supplemented kindly by the Telecooperation Group at Darmstadt University of Technology[101]. The crawling process took approximately ten days.
In a second step, the check-ins where linked to observations so as to have the smallest temporal difference between check-in and observation. Stations report either hourly, every half hour or every fifteen minutes, so the distance was not too big.
The data obtained had a total of 32 variables. Of these, nearly half where the same measurement expressed in different scales, for example if the temperature was both expressed in Celsius and Fahrenheit. Table 18 gives an overview over the variables, their measurement units and availability.
As can be seen, four variables are reported by only a few stations, namely water temperature, visibility, solar radiation, and pressure tendency, the latter being the change of pressure over time. Wind-related variables on the other side are reported for most check-ins (around two third) and include wind direction, wind direction in degrees, wind speed and wind gust (the speed of short bursts of wind speed).
Pressure related variables are reported for nearly 90% of all observations and denote the pressure at different heights. As it is not plausible that there are sensors at all those hights, it can be assumed that pressure at 1500m and pressure at sealevel is derived from the pressure mesaured by the station.
Lastly, some variables are measured by all stations. These include temperature, wet bulb (a way to indicate humidity by calculating the maximal possible lowering of temperature through vaporizing of water), dew point (temperature at which water begins to dew), heat index, which combines temperature and humidity to determine the human-perceived temperature, and wind chill, which is similar, only that it combines temperature and wind speed for a felt air temperature.
In summary, there are many variables which can be used to describe the environmental weather conditions under which a check-in is performed. Unfortunately, these variables do not include perspiration data, which could potentially be a good indication for a user's behavior. The reason for this may lie in the fact that focus for MesoWest lies more on variables describing wind, temperature, moisture and pressure than on precipitation.[102] When taking a look at the values itself, it seems that the quality of data available is an issue for some measurements. For 10% of the values from temperature and wet bulb, temperatue in degree farenheit is 0, which is an temperature of -18 degrees celsius. While not being an expert in those fields, it seems that those values are not corresponding to acutal temperatures, but are default values entered into the database.
Additionally, dew point, heat index and wind chill often have values that seem to be default values. This is especially obvious for wind chill, which is computed using temperature and wind speed and which also has values even if no wind speed is indicated.

[101] http://www.tk.informatik.tu-darmstadt.de/
[102] http://mesowest.utah.edu/html/help/faq_awips.html

Variable	unit(s)	availability
temperature	°c/°f	100%
wet bulb	°c/°f	100%
water temperature	°c/°f	1.7%
relative humidity		89%
wind direction	-	77.7%
wind degrees	degrees	77.7%
wind speed	kt/mph	55.4%
wind gust	mph/kt	61.5%
pressure	in/mb	89.4%
pressure 1500m	in/mb	89.4%
pressure sealevel	in/mb	89.4%
altimeter	in/mb	89.4%
dewpoint	°c/°f	100%
heat index	°c/°f	100%
wind chill	°c/°f	100%
pressure tendency	mb	9%
solar radiation	w/m^2	<1%
visibility	km/mi	<1%

Table 18: Variables, their units and availability for MesoWest

5 Experimental Setup

In this section, we describe the experimental setup used to test our hypotheses. Section 5.1 explains the approach chosen for evaluating the first, which states that the addition of knowledge about a spot enhances POI recommendation for users. Apart from baseline features, we introduce four different feature groups, data imported from LGD and derived from Gowalla, which can be combined to express knowledge about a spot. How we convert implicit into explicit ratings and create test and training sets for each users is discussed afterwards. The classifiers used for learning preference models, namely Naive Bayes and Ripper, are presented in the last part of the section.

Part two discusses our approach for testing the second hypothesis, which states that adding context information to a check-in enhances recommendation of spots for users. Here, we first discuss different context-based recommender approaches, determining which one is suited best for our situation. Following this, the method we apply, based on contextual post-filtering, is discussed.
The last section addresses the approach for evaluating the experiment. Conducting an offline experiment, we point out its properties compared to other approaches, and illuminate how we sample users as well as their test and training sets. Finally, in the last section, we elaborate which approach is most suitable for measuring the impact of different feature groups and contexts on the quality of recommendations for our setup.

5.1 Content-Based Recommendation

In Section 4, it was described how we enhanced existing spot data by two means. First, we matched nodes from LGD with spots from Gowalla. For all spots with a matching node, we used FeGeLOD to automatically create additional features, based on the matched node's attributes.
Second, we modeled the environment of a spot with data from LGD. We introduced features describing the types of nodes in the vicinity of a spot. Compared to the node data, this approach had the advantage that such an environmental description could be created for every spot, not only those with a matching node.

Based on the existing Gowalla data and additional features obtained from LGD, we created different feature groups, listed in Table 19.
The first group, named "spotfields" with ID zero, is the baseline feature group, containing features derived from available spot information in Gowalla. These are derived from most of the data available at table "gw_spotFields", as described in Section 3.1, excluding data that is not related to the spot itself, such as information regarding the crawl, but, if available, including the category of a spot from table "gw_spotcategory". The free text spot descriptions are converted for every user individually into a bag of words representation, using an uni-gram tokenizer. This process leads to a total of 15 fixed variables, plus an average of 5.3 additional variables for the bag-of word-representation.
The second group, having ID one and called "friend features", is derived from existing Gowalla data for every user individually, describing the association of the user's friends with a spot. In this group, one feature describes the quota of a spot's check-ins performed by the user's friends, while the second feature describes whether the spot has been created by a friend of the user. This features can thus be considered a hybrid extension for a content-based approach, as collaborative features are created, with similarity between users defined by their friendship. It is introduced to make additional use of available Gowalla information and because friend connections are an important aspect of LSNs, enabling us to draw a connection between the behavior of a user and his friends.
The third group, named "Gowalla environment", describes the environment of a spot through the categories of spots that are in his direct vicinity. This approach is analog to the one used for creating LGD environment information for a spot and serves to compare these two environment descriptions. It consists of 388 features, one for each category.

The next group with ID three, named "LGD environment", consists of features based on a spot's environment in LGD, as described in Section 4.3. This group has a a total of 5314 features, each one describing one LGD type.
The last group, with ID four and name "node data", consists of features obtained through the use of FeGeLOD for spots with a matching node, with missing values in case none was found. Of these features, a description of a place's wheelchair accessibility in free-text, was converted to a bag-of-words representation, again with an uni-gram tokenizer. A total of 342 features, plus the bag of word representation for the wheelchair-accessibility description with an average number of 8.67 features, are in this feature group.

ID	Group	Features	#features
0	spotfields	data available from Gowalla database	16+5.3
1	friend features	features describing the quota of friends of a user that checked-in at a spot and if a friend created this spot	2
2	Gowalla environment	modeling of a spots's environment through categories of Gowalla spots in the vicinity of spots's position	388
3	LGD environment	modeling of a spots's environment through the types of LGD spots that are in the vicinity of spots's position	5314
4	node data	features of LGD node matched with Gowalla spot (if available)	342+8.67

Table 19: Feature groups, with their ID, name, a description and the number of features

Based on these groups, training and test data for every user is assembled. With four different feater groups that can be added to the baseline group zero, we have a 2^4 full factorial design, with four factors and two levels, leading to sixteen possible combinations.[103]

With spots described as feature vectors, we can create a profile of user preferences by presenting examples of a spot's feature vector along with the user's rating for the spot to a classifier. As we have no explicit user ratings available, the user's preferences for a spot has to be derived from the number of times he visited it. Such implicit ratings are usually converted into binary or scalar explicit ones by discretizing visiting frequencies in some fashion and use these discretized values as ratings given by the user.[104] For a binary rating, an example would be to take the user's average number of visits per spot as a threshold, assigning to every spot with less visits a negative (dislike) and to all spots with more visits a positive (like) rating.
A disadvantage of such an approach is that it creates ratings that may be distorted for location-based items using content-based recommendation. Consider the example that a user travels around, often visiting different coffee shops of the same chain only once and, while being at home, visiting his work place and other spots often. As he visits many coffee shops once, with the transformation described above, they would be classified as "dislike" as they are probably below the average. Especially for content-based recommendation, this would mean that features common to these coffee shops, like the name, would be associated with a negative rating, which is obviously not desirable as he seems to like the chain.

One way to avoid such problems is by using spots the user did not visit as negative examples for the classifier. For this approach, the idea is that a user considers a set of spots from which he choses to check-in at one, rejecting the others. As we do not know which spots the user considers prior to checking-in, for LSNs, an approximation can be to assume he considered spots that are in the vicinity

[103] A factorial design is an experimental design where different factors (feature groups) can be applied at different levels (in our case two: include/not include). To view our setup as a factorial design will be beneficial for analyzing the results. See [43, p. 162ff] for an introduction.
[104] See [17, p. 23ff.] for examples

of the one he finally settled for. Assuming that he is aware of all possible spots in the area, the spot he checked-in then constitutes a positive training example, while all other spots the user did not visit in the area are negative training examples. For the example above, this would mean that every check-in at the chain would be a positive example, associating the features of the coffee shops with a positive classification and the spots in the area he did not visit are negative examples.
In our setting, we approximate "area" by considering all spots that are one kilometer around the checked-in spot as candidates, which resembles a walking distance. For every check-in in the training set, we then present the checked-in spot as an example of a positive rating "Visited" (1) to a classifier and all other spots in the area one kilometer around the checked-in spot as a negative rating "Not Visited" (0). Seen from the perspective of recommendation system theory, this approach employs a form of contextual pre-filtering, as the spots considered for learning and evaluating are pre-filtered, based on location.

This is certainly only an approximation of the user's decision and has some shortcomings. First, we make the assumption that the user is aware of all spots that are in the area he is visiting, which is arguably often not the case. Second, defining "vicinity" as one kilometer around a spot where a user checked-in is probably inaccurate, as spots that can also be considered in the "vicinity" of another spot may be outside of this area, for example spots farer away, but quickly reachable by public transport. Addressing these issues would mean to define "vicinity" as perceived vicinity by the user, possible including driving distance, which is feasible, but out of scope for this thesis. Lastly, the spots in the vicinity of a check-in might not resemble the spots a user considered, for example when he wants to go to a museum and considers every museum in town or even state. But as we have no information about the users' decision process available, for example which spot pages a user looked at or for what keywords he searched in the LSN, his considerations can only be approximated.
For evaluation, the same approach can be chosen. For every test check-in, all spots around the one where the check-in was performed are presented to the recommender. Then, the system generates a list of recommendations (spots classified as "Visited") to the user. Such an setup closely resembles how location-based services (i.e. the recommender service of LSNs) work, as the answer to a request for recommendations is adjusted to a location (the user's location or one denoted)[105].

For classification, we choose two different approaches. The implementations used are taken from Weka, a machine learning suite.[106]
The first one is the use of a Naive Bayes classifier [35, p. 338-345], a popular choice for content-based recommender systems, that has been used in many applications [53, p. 53,92]. It is a probabilistic classifier which computes the probability of an item belonging to a class C, given its feature vector $F_1, ..., F_n$, i.e. $p(C|F_1, ..., F_n)$. According to the Bayes-Theorem, this can be written as:

$$p(C|F_1, ..., F_n) = \frac{p(C)p(F_1, ..., F_n|C)}{p(F_1, ..., F_n)} \tag{16}$$

Assuming conditional independence of the features, this equation can be simplified to:

$$p(C|F_1, ..., F_n) = \frac{1}{Z}p(C)\prod_{i=1}^{n} p(F_i|C) \tag{17}$$

where Z is a normalization factor which assures that, given a feature vector, the sum of the probabilities over all classes is one. The a-posteriori probabilities for a feature given a class are estimated by using a maximum likelihood method for nominal values and a assumption of normal distribution for numeric

[105] The Foursquare Recommender Engine for example requires to specify a place or the user's coordinates for recommendation `https://developer.foursquare.com/docs/venues/explore`
[106] `http://www.cs.waikato.ac.nz/`

attributes.
Although the Naive Bayes assumption of class-conditional attribute independence is clearly violated (for example the name “Starbucks” and the category “coffee” are not independent), such classifiers perform very well in classification tasks. [51, p. 338]

The second classifier used is a rule based classifier called Ripper [63], which is closely related to decision trees. In the literature on content-based recommender systems, it is reported that decision trees have a tendency to base a rating on as few tests as possible, which can be an disadvantage if there are many relevant features. Ripper is said to perform competitively with other recommender systems. This is attributed to its post-pruning algorithm, thus making it usable for our purposes. [20, p. 332] The implementation used in Weka, called JRip, has some small changes compared to the original implementation, as described below.
The algorithm can be divided into multiple stages. In the fist stage, called building stage, rules are created. For this, the training data is divided into a growing set, on which basis rules are created, and a pruning set, used in the following to shorten the rules. We use the default setting for JRip, which assigns two thirds of the training instances for growing and one third for pruning.
In the first phase, the grow phase, rules are “grown” by greedily adding antecedents until a rule is 100% accurate. In every step, every possible value from every attribute is tried, chosing the one with the highest information gain.
In the second phase, called pruning phase, each rule created is pruned of a final sequence of antecedents. This sequence is chosen by maximizing the quota of positive examples in the pruning set covered by the rule versus the total number of examples in the pruning set.[107] Those steps are repeated until either the description length is 64 bits greater than the smallest description length met so far, if there are no positive examples uncovered or if the error rate is greater than fifty percent.
The next stage is the optimization stage. Here, for every rule generated in the first stage, two variants are created, using the same approaches as in the building stage, but with random data. One variant is generated from an empty rule, while the other is created on the basis of the original rule. The pruning metric here is the quota of true positive and false positive examples and all pruning examples. From these three variants, the one with the smallest possible description length is picked for the rule set.
If, after performing these steps for every rule, there are still positive examples that are not covered by a rule, more rules are created using the building stage. Finally, those rules are deleted from the rules set that would increase description length if they where in it.

5.2 Contextual Information

With the data acquired from MesoWest,two contexts, weather and time, are available for context-based recommendation. In Section 4.4, we described how weather information for the time and place of check-ins were acquired. Time information, i.e. the date when a check-in occurred, is part of the check-in records of the Gowalla dataset. As its timestamps are in UTC, it is first necessary to transform them into local time to make use of them. This is done by asserting the timezone of a spot, and then converting its check-in timestamps into local time.[108]
As described in Section 2.2.4, contextual information can be integrated in a recommender by various ways.
With contextual pre-filtering, items and ratings used for generating a model and/or making recommendations are filtered, depending on the context. For example when applying pre-filtering on a training set using the context of time, multiple models are created, where for generating each model, only check-ins performed at the corresponding time segment are used. One can for example create two models, one for weekdays and one for weekends, describing different preferences of a user for these two time windows.

[107] The rule used in this implementation differs from the one described by Cohen, which maximizes positive minus negative examples divided through positive plus negative examples covered by the rule
[108] Information about polygons of timezones were acquired from `http://efele.net/maps/tz/world/`

One could also pre-filter spots by including only those in the vicinity of a checked-in spot as negative examples if it is probable enough that the user considered it, for example not adding parks as negative examples if it was heavily raining at the time of check-in.[109]
With contextual post-filtering on the other side, recommendations made by a traditional recommender are adjusted in dependence of the context. In our situation, one can for example change the recommendation for a shop, re-classifying it from "Visited" to "Not Visited", because a check-in occurred in the middle of the night, and the contextual data indicates that at this time of day, a check-in at a shop is highly unlikely.
Lastly, when using contextual modeling, a traditional recommender is extended to a multidimensional approach, incorporating additional context dimensions. For this, new models for recommendation have to be created, for example by extending a support vector machine with additional contextual dimensions.

The different approaches all have their advantages and disadvantages. First, both experimental pre-filtering and contextual modeling require us to generate new models for users, depending on the context. For pre-filtering, either multiple new models for context segments or one new model with pre-filtered data has to be created, which constitutes a considerable computational effort. Furthermore, when creating multiple models, first the context space has to be segmented in a meaningful way. Segmentations can be created by a domain expert and/or by automatic means. In the latter case, additional computational effort is necessary to find meaningful segmentations. In the literature, only very few (usually two) segments are created by domain experts, for example weekdays and weekends for the context of time. For us, this is not desirable, as a lot of potentially interesting context dimension would remain unused.
For contextual modeling, a single new model has to be generated, but as it incorporates at least one contextual dimension, one can expect that the complexity of the computations will rise. If for example time dimensions were added to a SVM, the additional dimension would likely make the fitting of the hyperplane more complex and thus more costly computational wise.
Post-filtering on the other side enables us to take the results from the recommendations already performed and adapt them to the context by changing the classification of an item if its classification is unlikely, given the context. This means that no new models have to be generated, which saves a lot of computational effort. On the other side, we first have to create probability distributions of a context, for which we need meaningful segmentations not for the context dimension space, but the check-ins, seen as a paring of user and spot. For example we could create probability distributions for individual spots (probability that a user will check-in at café "Rosemarie" giving the prevailing weather situation) but also for whole spot categories (probability that a user will check-in at a café giving the prevailing weather situation). In addition, this can also be done for users (probability for a check-in at a café for all users or for male and females users separately).
Considering all these factors, we decided to use contextual post-filtering as our strategy. It gives us the opportunity to use recommendations already performed by our content-based recommender. Also, different contextual dimensions can be aggregating into one probability estimation, describing the probability of a check-in, given the prevailing context. The particular post-filtering approach we use is based on Panniello et al. [47]. They introduced post-weighting, where ratings are multiplied with the probability of a check-in of a user at a spot, given the context, and post-filtering, were a spot is not recommended if the probability for a check-in given a context is too low. As we have only two classifications (1: "Visited", 0: "Not Visited), we used an approach similar to post-weighting. A recommended spot will not be re-classified and not recommended if the probability of not visiting the spot is higher than a threshold and a not visited spot will be recommended if the probability of a spot being visited is higher than a threshold.

As already noted, this makes it necessary to create probabilistic models of a user checking-in at a spot given the prevailing context. Such models can be generated with respect to different groups of users and spots, where the grouping can be done on the basis of every feature of users and/or spots.

[109] Here, the user would not reject the place in general, but not consider it given the prevailing weather conditions

When choosing which grouping of users and spots should be used for creating models, two aspects need to be considered. For every group, enough data should be available to create meaningful models for every combination of user and spot groups. Further, the groups should be meaningful, in the sense that it has been shown or it it is common sense that meaningful patterns will emerge. For example, grouping users by their last names and creating models for every one of those groups will probably not create meaningful models, as a connection between a user's behavior given a context and his last name is implausible.
In our opinion, there seems to be no promising segmentation, as only few information about the users are available, which do not go beyond very basic information. A possibility for further studies would be to create groups based on the behavior of users, for example separating early birds and late risers or people going outside despite adverse weather conditions and people staying in. Also, in the literature reviewed, there were no findings on which we could base a segmentation of users for contexts of time or weather. For this reasons, we do not create models for different groups of users.
Learning context models for individual spots on the other side seems to be more promising, as more information on which basis grouping can be performed are available. Although it would be possible to create models for individual spots, which would probably be the most accurate models, given the power-law nature of check-in frequencies, many spots have only very few check-ins, making it hard to generate meaningful models for them.
Because of these considerations, we decided to create models for the spot categories of Gowalla in both contexts. As described in Section 2.4.1, it has been shown that check-ins at spot categories underly different temporal patterns. Something similar could be assumed for the context of weather, with different categories of spots (parks, cinemas) having different check-in frequencies depending on the weather, for example parks being less visited and cinemas being more visited if it is rainy. Additionally, as, compared to the number of check-ins, only a few categories exist, meaningful models can be created for every category due to the sufficiency of data. Note that other possibilities may be considered for grouping, for example learning models for different categories of spots in different countries or other spatial entities. Finding the optimal segmentation is a task that is beyond the scope of this thesis, but poses an interesting research challenge, especially as spatial data gives a wealth of possibilities.[110]

To get a probability estimation from a context feature vector, we use the same approach for both context domains. Essentially, we use a procedure similar to a Naive Bayes classifier, but with the modification of an equal-distributed prior. The reason for this lies in the fact that for the probability of a check-in given a context, it is not essential how many check-ins have been performed at this category of spots. Again, the assumption of conditional independence of all context dimensions is violated, but it enables us to easily compute the necessary probabilities. For being able to compare them, we normalize the probability of a check-in and no check-in for a given category to one.
This means that for every category of spots, we compute the probability of a check-in given a context by multiplying the probabilities of each context dimensions while for the probability of a check-in not happening, we compute the probabilities based on all check-ins that were not performed at this spot category. The check-ins we used to create the models are the first 80% performed.
The context dimensions of both time and weather are discretized as follows. For time context dimensions, we manual discretize the dimensions, based on our own experience and the insights from literature reviewed in Section 2.4.1. We discretized the time of day hour-wise, created a dimension for every weekday, one for weekdays and weekends, one for the month and finally the day of the month. Thus the probability for a check-in to occur at a Monday is simply the quota of all check-ins occurring Mondays divided through all check-ins.
For weather data, we chose a different approach. As we are confronted with continuous data and with-

[110] For example segmentation by administrative regions, especially if for those regions additional data such as household income is available, topological regions or regions defined by user behavior, like countries with and without siesta

out any domain knowledge, we used a discretization strategy integrated in Weka, which automatically discretized a dimension based on the minimum description length.

5.3 Evaluation of Recommender Systems

The literature distinguishes between three different processes for evaluating a recommendation system: offline experiments, user studies and online experiments. [53]
Offline experiments represent the simplest way for evaluating a recommender system, as they require no interaction with real users. Such a process uses pre-collected datasets of users, items and their rating to simulation an online process. This is done by training a recommender with the majority of all transactional data, using the remaining part not presented to it as hidden knowledge how the user will rate items, which is used to evaluate the recommender. Such experiments require no interaction with real users, thus allowing to compare algorithms at low cost. The downside is that one has to assume user's behavior is modeled well by the existing data and the fact that the influence the recommender has on the user is not modeled, as it is only modeled if the recommender predicts the user's ratings well, not if the user follows and likes the recommendations made.
The second approach are user studies, where a number of test subjects are recruited and asked to perform several tasks requiring an interaction with the recommendation system. The test subject's behavior can then be observed and recorded and qualitative questions can be asked before, during and after the task is completed. This allows to observe the behavior of an user and the recommender's influence on his behavior, while also giving the ability to collect qualitative data. The disadvantage lies in the fact that those studies are expensive to conduct, both in terms of time and resources, which limit such studies to a small number of subjects. Also, the scenarios have to be repeated numerous times, further limiting the range of distinct tasks that can be tested. It should be taken into consideration that the test population should resemble the users of the real system as close as possible, and that the subject's behavior can be biased due to their knowledge that they are observed.
Finally, there are online evaluations, which try to measure the change of user behavior for different recommender in a real world application. Such systems employ an online testing system with multiple algorithms, which redirects the traffic of users to alternative recommendation engines and records user's interactions with the different systems. Thus, it allows to monitor the behavior of real users in a real environment, which is the most realistic scenario. But for such a test, a first necessity is that an actual real world system exists, which may not often be the case in an academic research environment.

In this thesis, we will perform an offline experiment with data obtained from Gowalla and its semantic additions obtained from LGD and MesoWest. For an offline system, a first step is to decide how the data available is separated into training and test sets. For this, different possibilities exist. [53, p. 5f.]
First, one can decide if the training set should be a percentage of all transactions, or a percentage of every user's transactions. For collaborative filtering, the former is usually the case, as this means that a part of the user/item matrix serves as training data. As for a content-based recommender system, profiles are generated for each user independently, the latter approach is more appropriate, as it ensure that for every user, the same quota of their check-ins is available for training and testing.
Second, test data we have available per user can be brought into a temporal order, i.e. a sequence of check-ins done by the user. This gives us the choice between sampling test and training data randomly from a user's transactions, or take advantage of the temporal order, with the training data being the first k percent of transactions in the temporal order and the test data being the last $100 - k$ percentage. As the second approach resembles more closely real-world applications and no evident advantage can be found for the first method, we will keep the temporal order of transactions per user.
Lastly, we have to decide on the quota between training and test data. We decided to use 80% of the data as training data and 20% of the data as test data, which seem to be the predorminant quota in the literature reviewed.

Additionally, a more extensive approach is possible. One could create a new preference profile for a user after every j transactions[111], which would be used to evaluate the next j transactions. This would simulate the evolvement of the user profile and give insight on how and if the system improves with increasing use. Considering that we already have a large number of models per user and the fact that we are more interested in the question if the addition of data improves recommendation at all, we refrain from such an approach.
For evaluation, we sampled two experimental groups. The first one represents the average active user of Gowalla. It consists of 1000 randomly sampled users with at least 50 check-ins. The second experimental group represents users that often check-in at spots for which a matching node in LGD could be found. Here, we too randomly sampled 1000 users with at least 50 check-ins, but with the additional requirement that at least 50% of their check-ins were performed at spots with a matching LGD node.

Lastly, we have to address the issue of how to evaluate the recommendations given made for a test check-in. When performing offline experiments, prediction accuracy is a frequently seen way for evaluating the performance of a recommender system, where it is measured how closely the recommendations resemble a user's real preferences.
Prediction accuracy can be divided into three broad classes, namely measuring the accuracy of rating predictions, measuring the accuracy of usage predictions and measuring the accuracy of rankings of items. [53, p. 15ff.]

Measuring the accuracy of rating predictions measures how close the rating predicted by a system resembles the actual rating of the item by the user, in our setup if the spot where the user checked-in was rated as "Visited" (1) and all other spots rated as "Not Visited"(0). Metrics used to measure the prediction accuracy are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).
In our case, if a check-in is evaluated, the checked-in spot and the ones in its area are considered for recommendation, forming the set T. For the spot a user checked-in, he assigns the rating $r_{ui} = 1$ to the visited spot and $r_{ui} = 0$ to others in T. When recommending, the system generates a predictive rating $\hat{r}_{ui}$ for user u and every item i in T. RMSE and MAE can then be computed by:

$$RMSE = \sqrt{\frac{1}{T}\sum_{(u,i)\in T}(\hat{r}_{ui} - r_{ui})^2} \qquad (18) \qquad\qquad MAE = \sqrt{\frac{1}{T}\sum_{(u,i)\in T}|\hat{r}_{ui} - r_{ui}|} \qquad (19)$$

Compared to MAE, RMSE penalized large errors more heavily due to the squared difference of predicted and actual rating. These two metrics can be varied in various ways, for example by computing them for every user separately, giving an indication the error for individual users.
In our case, when using rating prediction, RMSE and MAE measures yield the same values, as the rating error is either one or zero. For our approach, such a measure of accuracy has a major drawback. As our aim is that for each check-in, only the one spot where the check-in was performed should be categorized "Visited", RMSE (and MAE) has the same value for the situation where all spots are classified "Not visited" and the situation where along with the visited spot, one not visited spot is classified "Visited" too. If more than one not visited spot is classified as "Visited", RMSE is even higher, compared to the situation where all spots are classified "Not visited".
This favors recommender systems that classify every spot as "Not Visited" over those which classify some not visited spots along the visited spot as "Visited". In our opinion, it is more desirable that some spots are recommended at all, even if along with the visited one, some rated by the user as "Not Visited" are recommended. Hence, RMSE and MAE is not suitable for our approach.

[111] In the most extreme case $j = 1$

Another way to assess prediction accuracy of a system is through usage prediction. Here, it is measured if the user "uses" the item, for example if a recommended spot has been visited or not and on the other side, if the spot has been recommended or not. For a check-in and its appendent recommendation, the possible outcomes shown in Table 20 are possible:

	Recommended	Not recommended
Used	True-Positive(tp)	False-Negative (fn)
Not used	False-Positive (fp)	True-Negative (tn)

Table 20: Possible outcomes when item is considered for recommendation to user

Based on this, the following metrics can be computed:

$$Precision = \frac{\#tp}{\#tp + \#fp} \quad (20) \qquad Recall = \frac{\#tp}{\#tp + \#fn} \quad (21) \qquad FPR = \frac{\#fp}{\#fp + \#tn} \quad (22)$$

In our situation, these three measures express the following. Precision can be seen as the precision of a recommendation in the sense that the more irrelevant items along with the visited spot are recommended, the lower precision is. If the visited spot is recommended, then precision is one divided through all recommendations. If the visited spot is not recommended, precision is zero. Thus it is desirable that precision is as high as possible, expressing that the right item is recommended with few to no irrelevant items. Not that this rating is absolute regarding non-relevant items, meaning that the total number of items along with the visited spot is used to compute precision, disregarding the total number of items to be considered.
Recall can be either zero or one, stating if the visited spot has been recommended or not. Likewise, for recall a high value is desirable.
Finally, the false positive rate describes how many irrelevant spots have been recommended, regardless if the visited spot has been recommended or not. Here, a low value is desirable, as it indicates that only a few irrelevant spots are recommended.
Another practice is to take the first n recommendations from all recommendations and compute precision based on this list, varying the list length n. With this, a precision-recall curve or an Operating Characteristic curves, the first emphasizing the proportion of recommended items that are preferred while the second emphasizing the proportion of items that are not preferred and that end up being recommended. For our approach using list lengths, is not feasible, as our rating is binary, and it is not possible to decide which items are part of the list if more items are recommended than the length of the list. If for example 12 items are recommended an the list length is five, it is hard to determining which items are on the list and which are not, distorting the results.[112]

A third way for measuring predictive accuracy is to use ranking measures. Here, the ordering of the recommended items based on their ratings is compared with a reference order, giving an indication of how similar the two are.
A problem using this measure exists when items are tied, meaning that either the system or the user rates items the same, making it impossible to bring them into a definite order. In our case, spots are classified into two categories, namely "Visited" and "Not Visited" and within these groups, a non ambiguous ordering cannot be given. For the reference rating, the same is true, as only the visited spot is classified "Visited" while all other spots are classified as "Not Visited".

[112] A possibility here would be to use the probability of the visited class as a measure of certainty of classification, ordering the spots in that order. Although a possibility, it has never been encountered in the literature, hence we will not use it.

For not penalizing if one item is ranked over another if they are tied in the reference ranking, the Normalized Distance-based Performance Measure(NDPM) can be used. If there are reference rankings r_{ui} and system rankings $\widehat{r}_{ui}$ of n_u items i for user u, we can define:

$$C^+ = \sum_{ij} sgn(r_{ui} - r_{uj}) sgn(\widehat{r}_{ui} - \widehat{r}_{uj}) \quad (23) \qquad C^- = \sum_{ij} sgn(r_{ui} - r_{uj}) sgn(\widehat{r}_{uj} - \widehat{r}_{ui}) \quad (24)$$

$$C^u = \sum_{ij} sgn^2(r_{ui} - r_{uj}) \quad (25) \qquad C^{u0} = C^u - (C^+ + C^-) \quad (26)$$

NDPM is then given by

$$NDPM = \frac{C^- + 0.5C^{u0}}{C^u} \quad (27)$$

In this formulas, C^+ are the ratings performed by the system that are consistent with the user's rating and C^- are those that contradict the user's order. C^u are the number of item pairs that are not tied in the user ranking and C^{u0} are the number of pairs that have a tie in the system's ranking. NDPM then expresses the number contradicting ratings plus half of those that are tied, divided through all possible rankings. NDPM takes a value between 0 and 1 where 0 means that the rankings are exactly the same, while 1 means that they contradict completely.
In our approach, this rating expresses the following. C^u is always the number of candidates minus one, as only the visited spot is not tied with all other spots. If the visited item is recommended, C^+ is the number of not visited items that are not recommended, while C^- is zero. If the visited item is not recommended, C^- is the number of not visited items recommended while C^+ is zero. Finally, C^{u0} is the number of items tied with the visited item, i.e. the number of items that have the same classification like the visited one. NDPM can actually be expressed as usage-based prediction accuracy metric. If the visited spot has been recommended, C^- is zero, and the remaining term is the same as half the false positive rate. If the visited spot has not been recommended, C^-/C^u is the false positive rate while $\frac{1}{2}C^{u0}/C^u$ corresponds to $1-FPR$. Uniting these two cases with the help of recall and simplify the terms, $NDPM = \frac{1}{2}(1 - recall) + \frac{1}{2}FRP$. Thus NDPM lies between 0 and 0.5 if the visited spot has been recommended, with the value being one half FPR, while if the visited spot has not been recommended, NDPM lies between 0.5 and one, the value being 0.5 plus FPR multiplied with 0.5.

Considering all these options, we will rely in our evaluation on usage prediction accuracy. The main metric we examine and we will try to enhance is average precision, because in our case, it represents a good indication of the quality of recommendation being influenced both by increasing recall and decreasing FPR. In order to tackle the case where no recommendation has not been made at all, precision is then defined as zero.[113] We will also report recall and false positive rate, as they give an indication how many not visited spots have been recommended as well as for how many check-ins the visited spot have been recommended or not.

[113] NDPM would also be possible, having the downside that it does not consider the absolute number of items recommended, but on the plus side differentiates between check-ins with recall zero and different FPRs, giving a clue about how "bad" such a recommendation is, whereas precision in this case is always zero.

6 Results

This presents the results of our experiments. In the first section, we evaluate the first hypothesis, namely that the addition of features to spots enhances recommendation. For every classifier, we evaluate the different feature group combinations under various conditions, and, if possible, make general statements about the impact of individual feature groups. Lastly, we analyze learn times for both classifiers.
In part two, we describe the results for context-recommendation, evaluating the second hypothesis stating that the addition of contextual information enhances the performance of recommendation. The evaluation is done for every type of context separately, again evaluating the performance changes for both classifiers under different conditions.

6.1 Content-Based Recommendation

In the following two sections, we will review the results for Naive Bayes and Ripper separately.
For both classifiers, we examine performance from two perspectives. First, evaluation is done including all test check-ins. This includes repetitions as well as check-ins from spots for which check-ins of the user exist in his training data. This set of check-ins and spots will be called “all spots” or “all check-ins”.
In order to see how well a user model can predict check-ins when confronted with yet unseen spots, we additionally examine the results if only check-ins performed at spots where no check-in was performed by the user that is part of his training data are considered. In the following, we refer to this group of check-ins and spots as “new check-ins” and “new spots”.
For each evaluation, the two experimental groups sampled can be used to examine the differences of two types of users. One group represents the average active user, while the other one represents an average active user that frequently visits spots with corresponding LGD nodes. Especially regarding the feature group of additional data from matched nodes, we hope to give insights on the differences if more spots have a corresponding node from which data can be imported.

Evaluation for a group of check-ins is done as follows. First, we compare metric values for the baseline model of both experiments, examining differences that are a side effect of our sampling process. Then, we take a look at the changes of metrics relative to the baseline model for different additions of feature groups, showing which combinations have the best impact and giving an indication how and if feature groups influence each other when found in the same model.
To see the impact of feature groups more clearly, we analyze their main effects and interactions, as described of percentaged changes relative to the baseline model[114], more formally by using the usual approach for analyzing factorial designs, as described by Montgomery et al.[43, p. 228ff.].[115] A feature group’s main effect is the difference between the average of a metric for the models with and without the feature group included. A significant main effect describes the average change the inclusion of a factor group causes, regardless of the model it is added to. Interactions describe if the value of an feature group’s effect depends on the level of one or more other feature groups. For example it might be the case that feature groups two and three interact, which means the effect for feature group two on a model depends on the presence or absence of feature group three and vice versa. For example it could be that the inclusion of both feature groups generally raises a metric’s value beyond the sum of the individual’s feature group main effects.
In the following, tests were usually done with the appropriate two sided t-test(depending on the applicability, one- or two-sided, coupled or uncoupled test). Stars indicate the level of significance[116] and if not significant, we denote the p-value.

[114] This allows us to compare the effects of the two experiment in a pooled regression with a dummy variable for the experiment. See Kleinbaum et al. for details [37, p. 221]

[115] Instead of ANOVA, we fit a full model, were the coefficients are half the effect’s value and test them for significance in the model. See Box et al. [18, p. 394f., p. 188] for details

[116] $* : p \leq 0.1; ** : p \leq 0.05; *** : p \leq 0.001$

6.2 Results for Naive Bayes

This section describes the results when using Naive Bayes for building preference models. Results for all check-ins are examined first, followed by the outcomes for evaluation with new check-ins only. In each section, after analyzing and comparing average precision, recall and FPR for the baseline models, we describe changes in percent relative to the baseline model for the different features combinations. The examination of main effects and possible interactions form the last part.

6.2.1 All Check-ins

The metric values of the baseline models of experiment one and two can be seen at Table 21. Check-ins from experimental group two have on average a higher precision and recall, along with a lower FPR, leading to the conclusion that recommendation for users from experiment two is on average better than for users from experiment one.
This might have its cause in inter-group differences, which are a by-product of our sampling process. A user sampled for experiment one performs on average 1.8 check-ins per spot, while this value is higher with 2.4 check-ins per spot for a user from experiment two. The average number of visited spots per user is with 78 for experiment two lower than for experiment one with 128, and this difference is the same if test and training sets are analyzed separately. As models trained and the tests performed for users from experiment two are done with fewer different spots that are visited more frequently, giving good recommendations is easier for experimental group two, as the models do not have to capture the wider variety of behavior shown by users from experimental group one.

Metric	Experiment 1	Experiment 2	Significance
Precision	0.079	0.102	***
Recall	0.367	0.440	***
FPR	0.201	0.146	***

Table 21: Average Precision, Recall and FPR for baseline models of Naive Bayes with all check-ins for experiments one and two

The impact of different feature combinations, expressed through percentage changes for each model relative to the baseline is displayed in Figure 16.
For average precision, changes are mostly positive and significant, the only exceptions being models 01 and 012 from experiment two with a highly significant negative change of 1.96 percentage points and a non-significant decrease of 0.49 percentage points. The increases in average precision for all other models are usually larger for experiment two, with the only exception being models 01, 012 and 013. If feature group four is included, the difference is particularly distinct between the two experiments. The strongest gain in average precision is achieved with model 01234 for experiment one with 7.45 percentage points and model 0234 for experiment two, with an increase of 13.69 percentage points, the highest gain for both experiments. From these observations, one gets the first impression that Naive Bayes in general benefits from additional feature groups in terms of higher average precision, and that for experiment two, which has more spots with additional LGD data, the benefit is stronger than for experiment one.
Taking a look at the change for average recall, gains are observed for models with feature group one and four, and again, the gains are particular strong if feature group four is included in experiment two. Models with feature group two and three on the other side seem to generally decrease average recall, if compared to the model without the feature groups. Here, a pattern seems to emerge, with feature groups adding knowledge about a spot itself lead to an increase in average recall when added, while feature groups describing a spot's environment lower average recall. Thus both for experiment one and two, the strongest increases is observed with model 014, which raises recall 7.21 percentage points for experiment one and nearly 20 percentage points for experiment two.

Lastly, taking a look at the changes in average FPR, the pattern observed for average recall is also present. Including feature group one and four into a model results in a higher average FPR, while incorporating feature group two or three into a model lowers average FPRs. Again, model four in experiment two triggers the strongest increases of all feature groups for both experiments. The best models in terms of decreasing average FPR are for both experiments model 023 with a minus of 6.58 percentage points for experiment one and 13.27 percentage points for experiment two.

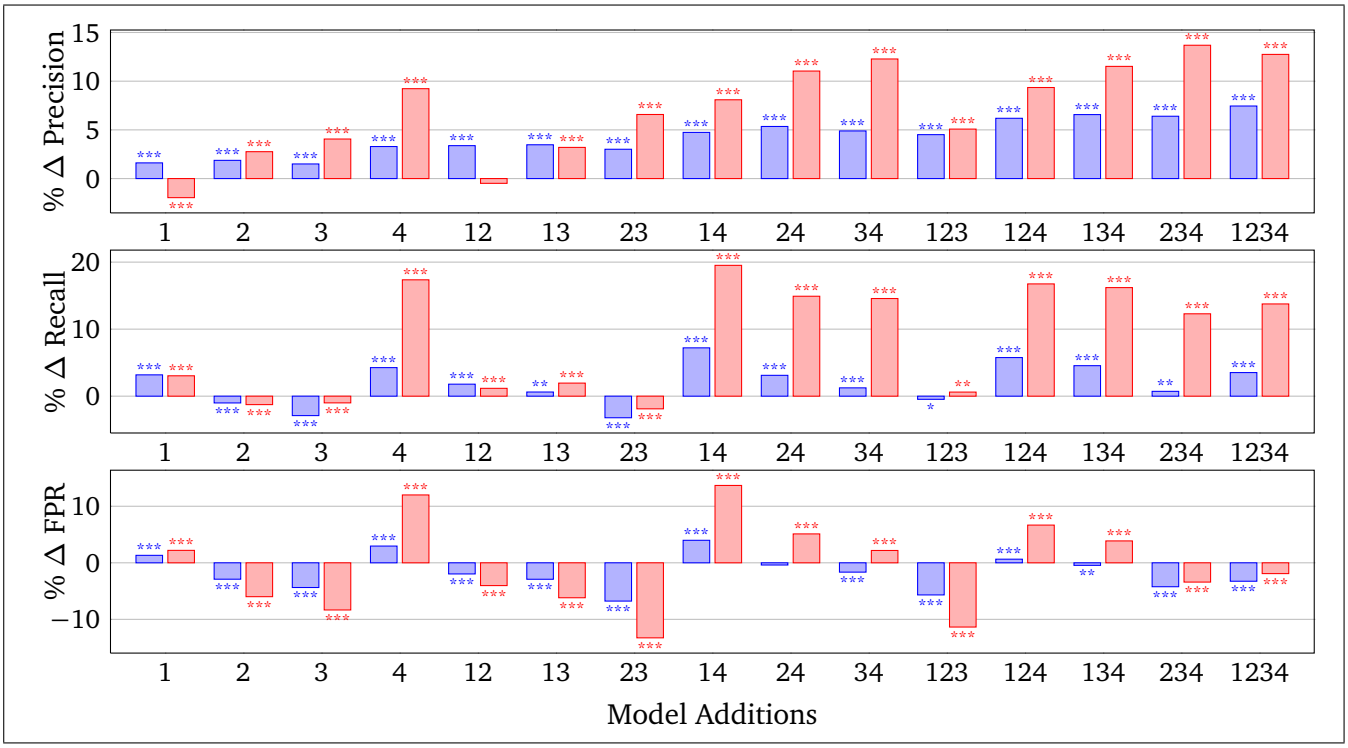


Figure 16: Changes relative to baseline model for precision, recall and FPR for all feature group combinations of experiment one and experiment two

Comparing different models with and without a specific feature group, one gets the impression that the effects of the groups on the three metrics are largely independent from each other. Adding a feature group to a model results in a a change that resembles the change of the initial model plus the change of the model with only the feature group added. Looking for example at average precision of model 023 in experiment one and adding the percentage change of model 04, the resulting bar resembles the one for model 0234.

To see the average effect of individual feature groups and significant interactions between groups, Table 22 displays main effects and interactions of percentage changes relative to the baseline model. We confined ourselves to only report significant interactions for reasons of space, unless one seems to be relevant in magnitude, despite its formal non-significance.[117]

As most metrics show significant main effect and no interactions, one can conclude that main effects affect metrics largely independent of each other, which is why we can examine them separately, allowing us to make statements on the influence of individual feature groups detached from particular models. Only for recall in experiment two, we have significant or nearly significant interactions between feature groups one and four, two and four and three and four. Although this usually means that one should not examine the main effects independent from each other, the impact of the interactions is small, compared to the sole effect of feature group four, which is the feature group interacting pairwise with the others. As it does not change the sign or general magnitude of the results, we nonetheless examine the effects on

[117] i.e. a p-value a little higher than 0.1. As it turns out, most interactions are of very small magnitude with a p-value beyond 0.7

recall for experiment two independently from each other, addressing the issue of interactions afterwards.

Taking a look at the values for feature group one, we see that the main effects on the three metrics are all significant. Although recall and FPR of the two experiments are similar and do not differ significantly, the differences for the effect on average precision is significant, with a positive impact for experiment one and a negative one for experiment two.
The question why for the two experiments, with similar changes in recall and FPR, the changes in average precision is so different, may be answered by the fact that the average size of the candidate list for experiment two is bigger than experiment one. Thus, with a similar increase in average FPR, absolute more spots are recommended on average for experiment two than for experiment one.[118] This in turn decreases average precision for experiment two more than for experiment one, as precision is defined with the absolute values of spots recommended, not the quota of all candidates. Although more check-ins have an recall of 1, and thus for them, precision becomes greater than zero, this positive effect on average precision seems to be unable to counterbalance the negative effect of more spots being recommended, eventually resulting in a lower average precision for experiment two.
Feature group two shows similar magnitude in percentage change and significance for precision in both experiments, as indicated by the fact that the two effects do not differ significantly. For average recall and average FPR on the other side, group two has better results for experiment two. The fact that precision is similar, despite recall and FPR showing different numbers, might be explained by the fact that for experiment two, recall and FPR both drop significantly stronger, counterbalancing each other's effects. Taking the number of spots in the environment of the checked-in spots for an indication of the richness of the spot features in the area, 3.8 for experiment one and 3.76 for experiment two is quite similar, resulting in the same drop in average precision, but different magnitudes of drops for average recall and FPR.
For feature group three on the other side, the change of average precision differs significantly, being nearly three times higher for experiment two than for experiment one. While the differences of the drops of average recalls in both experiments are not significant, for experiment two, average FPR is lowered twice the value of experiment one. An explanation for this difference is that spots visited by users from experiment two on average have 8.47 type descriptions in their environment, compared to 2.52 for experiment one.[119] This may make the description of a spot's environment more nuanced and results in not recommending more spots whose environment does not resemble the ones of visited spots, resulting in a higher increase in average precision.
Finally, feature group four affects all metrics for both experiments by significantly increasing precision, recall, but also FPR, yielding for all metrics in both experiments the highest increases of all feature groups. Comparing the two user groups, the effects are much stronger for experiment two, with the change in precision being nearly three times higher than for experiment one, the change in average recall being nearly four times higher, and for average FPR being more than four times higher than those of experiment one.Thus the inclusion of feature group four to a model increases average precision, with the effect being stronger for experiment two. As user from this experiment check-in on average three times more often at spots with corresponding LGD nodes, one can additionally state that this improvement grows stronger if more spots have matching nodes.

As already stated, there are significant or nearly significant interactions of feature group four with groups one, two and three. These effects basically state that if such a pair of feature group is present in a model, apart from the individual effects, an effect resulting from both group's interaction has to be considered. Note that this does not mean that an interaction effect 1234 has to be present, as this would imply that for example feature group four interacts with groups one, two and three, only if they are all present

[118] Exp. 1: 360.18, Exp. 2:419.79
[119] Again, we take the average number of types in the environment of the checked-in spot as a proxy for the general richness of the area's spots' environment descriptions

which is not the case.

	Experiment 1			Experiment 2			Inter Exp.
Effect	Precision(%)	Recall(%)	FPR(%)	Precision(%)	Recall(%)	FPR(%)	Significance
1	1.45*	3.00***	1.09**	-1.51**	2.25***	1.80 ***	***/ - / -
2	1.52*	-1.00**	-2.85***	1.79*	-1.91***	-5.85 ***	- / ** /***
3	1.42*	-2.53***	-4.03***	3.89**	-1.87***	-8.36 ***	***/ - /***
4	3.19***	4.04***	2.54***	8.59***	15.34***	10.46***	***/***/***
14	-	-	-	-	$-0.48^{0.12}$	-	- / - / -
24	-	-	-	-	-0.57*	-	- / - / -
34	-	-	-	-	-1.05***	-	- / ** / -

Table 22: Main effects and interactions for average Precision, Recall and FPR of experiment one and two for Naive Bayes and all check-ins

If one of the interactions appear in a model, the resulting average recall is influenced by it, changing average recall of all affected models on average by the value of the interaction. As one can see, the interactions present lower recall between one half percent to around one percent. This means that when for every model incorporating model three and four, apart from the main effects, the interaction between the two feature groups affect recall significantly, lowering the sum of all effects 1.05 percentage points. Such interaction likely come from the fact that feature group four has such a large impact on average recall, more likely influencing the value of many check-ins which are also affected by the other feature groups.

In summary, the best combination of feature groups for an experiment with respect to average precision can be derived from the performance of individual feature groups can be both derived from analyzing the impact of models as well as the general impact of individual feature groups. For experiment one, the optimal feature model is hence 01234, with an increase of 7.45 percentage points in average precision, an increase of 3.52 in average recall and an decrease of 3.26 in average FPR.
For experimental group two, the best improvement for average precision is 13.69% using model 0234, leaving out feautre group one, which showed an decrease in average precison. For this combination, average recall increases 12.29%, while average FPR decreases 3.41%.

6.2.2 Naive Bayes with New Spots

After analyzing the results for all check-ins, this section discusses results for Naive Bayes when only new check-ins are considered. Of all test check-ins, twenty percent are counted as“new”, i.e. check-ins at a spot where no check-in occurred that is part of a user’s training set.
Table 23 displays the results for the baseline model of both experiments. Compared to the results for all check-ins, both average precision and recall are generally lower for new check-ins only.[120] Average FPR shows smaller differences, especially for experiment one, with a small increase of one percentage point for experiment group one and 19.4 percent points for experimental group two.
The significant differences between the two experimental groups already observed for all check-ins are also present for new check-ins only, with precision and recall significantly higher and FPR being significantly lower in experimental group two. The differences have become smaller though, the differences in average precision for new check-ins being one third of the difference for all check-ins and for average recall and average FPR this difference is 68% and 53% respectively. These results are not surprising, as it is harder for a recommender to appropriately classify spots it has not encountered before.
Next, we take a look at the changes of metrics introduced by different feature groups, described in

[120] Precision: 58% of full evaluation’s average precision for experiment one and 49% for experiment two; Recall: 66% of full evaluation’s average precision for exp. one and 62% of experiment two’s

Metric	Experiment 1	Experiment 2	Significance
Precision	0.045	0.050	**
Recall	0.243	0.276	***
FPR	0.203	0.174	***

Table 23: Average Precision, Recall and FPR for baseline models of Naive Bayes with new check-ins for experiments one and two (test for significance with a two-sided, unpaired t-test)

Figure 17. In general, the impact on average precision is smaller in its magnitude, with the best improvement being less than ten percent, where for the evaluation with all check-ins, improvements are up to 12 percentage points. Some models, especially those including feature groups two and three together, now have negative impacts on average precision. Also, models are often not significant in their impact, especially for experiment one. And, differing to the evaluation with all check-ins, experimental group two now benefits from the addition of feature group one. The strong differences between the experiments for models with feature group four is again present in this evaluation. The best models here are for both experiments model 014 with an increase of 5.29 and 13.92 percentage points.
Taking a look at changes in average recall, only some models have significant positive impacts, while for most models it is significantly negative. One observes the familiar pattern, with increases for group one and four and decreases for two and three. Additionally, average recall is decreasing much more stronger with models incorporating feature groups two and three while the highest increases are lower compared to the evaluation with all check-ins. This leads to the situation that for models with more feature groups, the negative impact of groups two and three are not always counterbalanced by the benefits from groups one and four, leading to drops in average recall. The best models in terms of average recall are again model 014 with an increase of 4.51 and 10.83 percentage points.

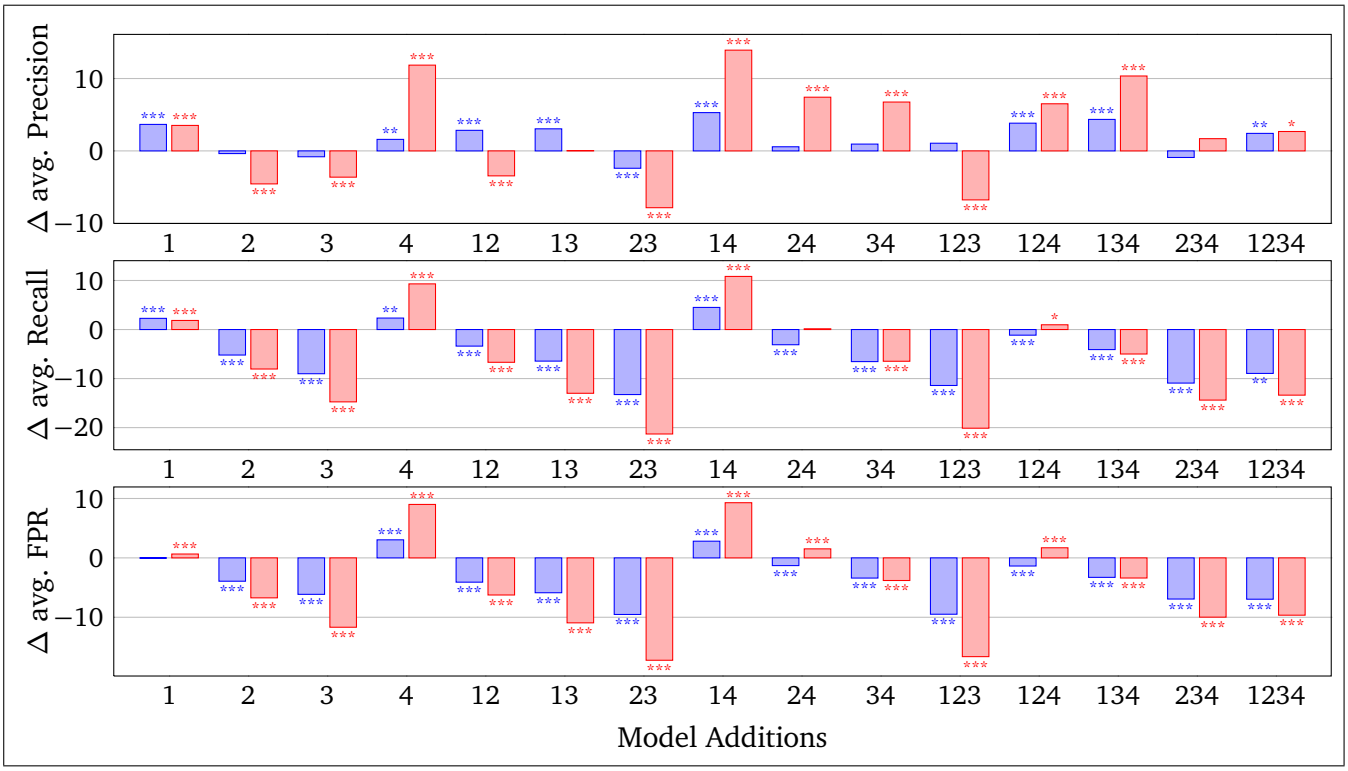


Figure 17: Differences of Precision, Recall, FPR for new spots and Naive Bayes for experiment one and two.

Lastly, taking a look at the changes in average FPR, we again see the familiar pattern of increases in

average FPR for groups one and four and decreases when groups two and three are added and again, the increases are stronger for models with feature group four than they are without that group. Here, decreases for average FPR are for some models stronger that the maximum decrease seen for all check-ins, and the best models are model 023 for experiment one and two with -9.53 and -17.24 percentage points.
The results show that the models with the highest increases are not the same for new check-ins than for all check-ins. To shine the light on the individual effects, again we take a look at main effects and significant interactions, described in Table 24. As can be seen, no interaction effects are significant, indicating that the effect of individual feature groups are independent of each other and can be analyzed that way.

For the first experiment, only feature group one has a significant positive effect on average precision. The gain apparently originates from a higher recall, as average FPR remains virtually unchanged. Feature group two and three even have a negative impact, albeit not significant, and feature group four has a positive impact less than a half of feature group one's effect, also being not significant. For average recall and FPR, we see effects similar to those observed for all check-ins, with positive impact of feature groups one and four on average FPR and average recall and negative impact for feature groups two and three on these metrics. For all other groups, negative and positive effects of average recall and average FPR seem to balance, resulting in no significant change in average precision.

	Experiment 1			Experiment 2			Inter Exp.
Effect	Precision(%)	Recall(%)	FPR(%)	Precision(%)	Recall(%)	FPR(%)	Significance
1	3.49^{***}	2.13^{***}	$0.004^{0.99}$	$1.89^{0.15}$	1.38^{*}	$0.45^{0.47}$	- / - / -
2	$-1.38^{0.22}$	-5.05^{***}	-3.73^{***}	-5.89^{***}	-8.20^{***}	-6.38^{***}	** /***/***
3	$-1.21^{0.28}$	-8.36^{***}	-5.68^{***}	-3.99^{**}	-14.60^{***}	-11.27^{***}	- /***/***
4	$1.38^{0.22}$	2.32^{***}	2.63^{***}	10.48^{***}	8.01^{***}	7.74^{***}	***/***/***

Table 24: Main effects and interactions for average Precision, Recall and FPR of experiment one and two for Naive Bayes and new check-ins

For experiment two, the effect of feature group one is also positive, though not significant. Again, we see a significant improvement in average recall, combined with a non-significant change in average FPR. Compared to the main effects for all check-ins, the change in average precision is thus positive because more visited spots get recommended, but on average not significantly more not visited spots. Like for experiment one, average precision for experiment two is negatively affected by feature groups two and three, but here, the effects are significant. Contrary to experiment one, feature group four has a significantly positive impact on average precision. For average recall and FPR, we see the familiar pattern with strong negative impacts of feature groups two and three, and positive impacts for feature group one and two.
Comparing these values with the ones for all check-ins, one can observe a pattern. While values for average FPR remain roughly the same for all feature groups and both experiments, the changes in average recall are either more negative or less positive, with the negative impact being some times higher than for all spots and the positive impacts being only a fraction of the values for new check-ins. Thus in general, it seems as fewer new checked-in spots are found, which is not surprising giving that these spots are new to the recommender, but that also that a similar quota of spots are recommended, instead of also recommending less.
For feature group one, the fact that average FPR does not change significantly is an interesting point. As the improvements in recall are significant, it seems to indicate that there is a strong connection between the behavior of a friend with respect to a new spot and a user's behavior and that it concentrates on the newly visited spot, not the ones in its surrounding. As it is indicated in the literature that a user has a tendency to go to places his friends go, one could conclude that a user specifically goes to new places his friends go and that this new spot stands out in this respect relative to his environment, as otherwise

more not visited places would have been recommended.
In summary, for new spots, one would use feature groups one and four, leaving out models two and three. The results seem to indicate that the environment features seem to perform poorly when confronted with new spots, a clue that it is not able to generalized enough to find patterns in the environment. As average FPR for the baseline models were similar to the one for all check-ins, the lower percentage in FPR changes for models with environment feature groups seem to indicate that they are responsible for over proportionally changing the classification of the visited spot to not visited. As this is not desirable, such variables should be either left out or, if possible, refined to yield better results.
Again, we see that for experiment one, average precision can be improved using feature group one. Furthermore, the increase in average precision for experiment one seems to indicate that this feature group is helpful when confronted with yet unseen spots and not just in cases were already visited places have to be recommended, underlining the good performance of this feature group. Feature group four seems to be able to also improve recommendation for new spots, and again, seems to yield better results if more node information is available.

Summarizing the results, one can see that Naive Bayes generally benefits from the addition of different feature groups when evaluating all check-ins. Differences between the experimental groups can be contributed to inter-group differences regarding the user's check-in behavior and friend connections. Feature groups describing the environment generally have an advantageous impact on FPR and a negative impact on average recall, while spot-related feature groups (feature group one and four) have an positive impact on average recall but also lead to an increase in average FPR. The best models combine these feature groups, balancing effects from both feature groups. Additional node information show the best improvements of all feature groups and, as the differences between the experimental groups show, becomes even better if more information is available.

6.3 Ripper

In this section, we discuss our findings when using Ripper as classifier for recommendation. Like for Naive Bayes, the first section analyzes how different feature combinations affect predictive accuracy for all check-ins while in section two, results regarding evaluation only with new check-ins are described. As Ripper is a rule-based classifier, a third section report rule statistics regarding different models and their rules created with this learner.

6.4 Ripper with all spots

Taking a look at Table 25 for results of both experiments when using Ripper with the baseline features, difference to Naive Bayes become evident.

Metric	Experiment 1	Experiment 2	Significance
Precision	0.067	0.067	-
Recall	0.153	0.170	***
FPR	0.053	0.054	-

Table 25: Average Precision, Recall and FPR for baseline models of Ripper with all check-ins for experiments one and two

Ripper has for both experiments a lower average precision, being 84 percentage points for experiment one and 45 percentage points for experiment two of the value for Naive Bayes. Precision does not differ significantly between the two experiments.
Taking a look at average recall and average FPR, Ripper shows lower average recall values, being just 41% (exp. 1) and 38% (exp. 2) of the average recall achieved with Naive Bayes. On the other side, Ripper has a much lower average FPR, being 27% and 37% of the values from Naive Bayes for experiment

one and two. Again, no significant difference between the two experiments regarding FPR is observed for Ripper. Thus, one can say that Ripper generally recommends less spots, both visited and not visited, and that the difference between the experimental groups, is generally smaller, possibly less affected from inter-group differences.

Taking a look at Figure 18 for the changes induced by the use of different models, one observes that most of them lead to a decline in average precision. The only significant exception is model 02 for experiment one, which increases precision 2.52 percentage points. Thus Ripper seems not only to barely profit from the addition of feature groups, but these additions tend to harm its performance.
For average recall, similar results can be seen, as only models 023 and 0124 for experiment one increase in average recall, while other feauture combinations lower it.

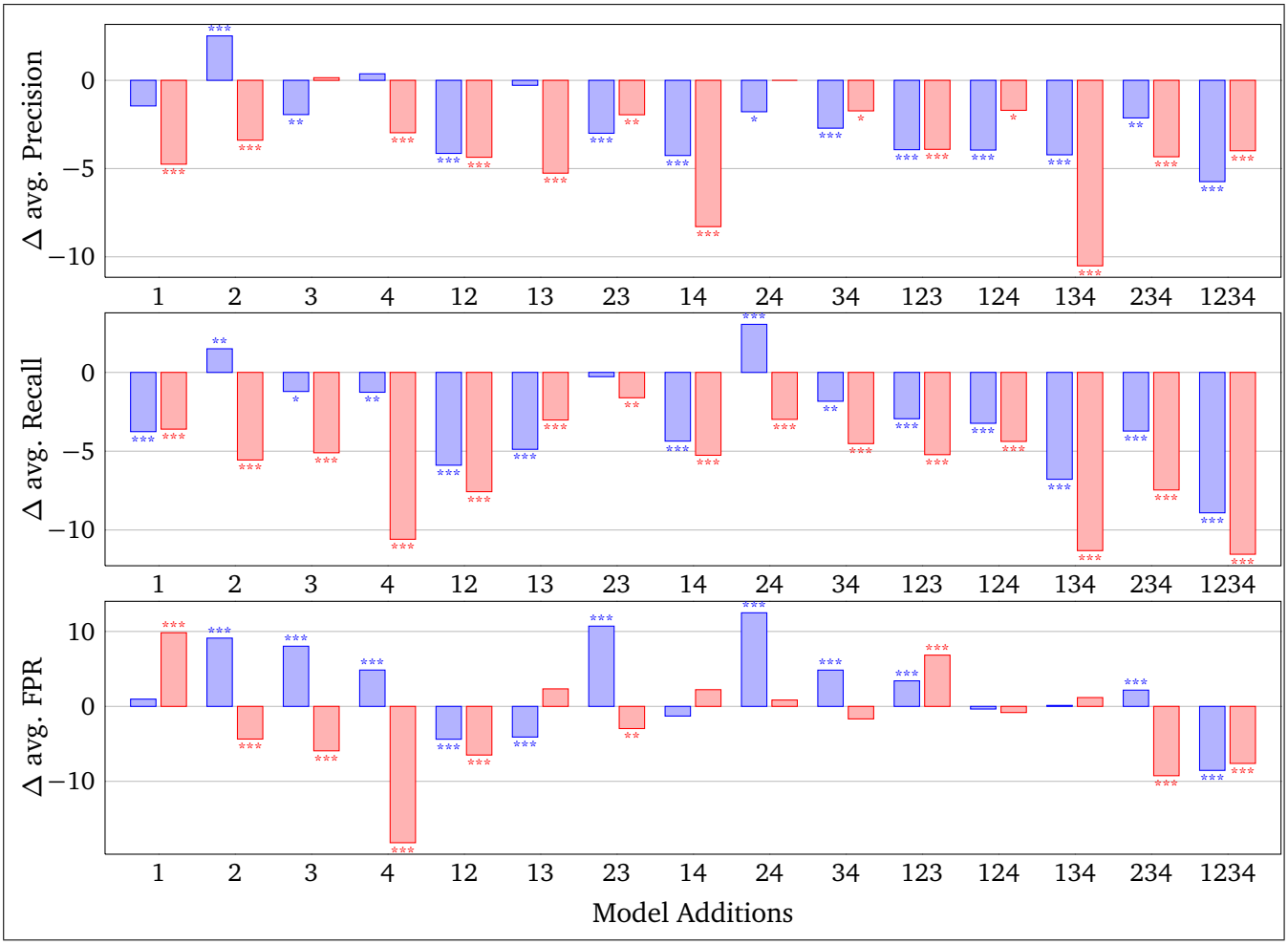


Figure 18: Differences of Precision, Recall, FPR for all spots and Ripper for experiment one and two

Average FPR on the other side is improved for some more models. The strongest decrease is seen for model 04, which lowers average FPR around 18 percent points, while for experiment one, the strongest decrease is the full model 01234 with -8.54 percentage points.

Main effect and significant interactions for Ripper are described in Table 26. For both experiments and all metrics, multiple significant interactions are present, which makes the analysis of main effects complicated. As only one model with a significant improvement exists, and as this model includes only one feature group, we pass on a in dept analysis of main effects and interactions.
An interesting point that can be seen though is that relying exclusively on the analysis of main effects for determining best feature group combinations can be dangerous. While only model 02 for experiment one

has a significant improvement in average precision, feature group two shows a negative and insignificant main effect and a significant negative interaction with feature group one. This is due to the different behavior of this feature group in different models, for example changing an insignificant deterioration of model 01 to a stronger, significant deterioration for model 012. Thus although on average the effect from feature group two is negative and insignificant, a model using only this feature group can nonetheless have a significant positive change in average precision.

	Experiment 1			Experiment 2			Inter Exp.
Effect	Precision	Recall	FPR	Precision	Recall	FPR	Significance
1	-2.87***	-4.73***	-8.27***	-4.06***	-1.76***	6.02***	- /***/***
2	-0.54$^{0.49}$	0.42$^{0.45}$	1.37*	1.70**	0.36$^{0.54}$	-1.66*	* / - / **
3	-1.84**	-2.15***	0.71$^{0.39}$	-0.27$^{0.75}$	-1.23**	-0.001$^{0.99}$	- / - / -
4	-1.18$^{0.13}$	-1.39**	1.36*	-1.74**	-3.30***	-4.0***	- / ** / **
12	-1.53**	0.85$^{0.13}$	2.74***	2.0**	1.0*	-4.14***	***/ - / -
13	1.54**	-	-	-	-1.34**	-	** / ** / -
23	-	-	-1.52*	-	-	-	- / - / -
14	-	-	-	-	-	-	- / - /***
24	-	-	2.17***	1.67*	1.69***	-1.58*	- / ** /***
34	-	-1.67***	-3.58***	-1.63*	-1.68***	-	- / - /***
123	-	0.80$^{0.14}$	2.39***	-	-	4.22***	- / - / -
124	-	-	-	-	-	1.58*	- / - / -
134	-	-	1.29$^{0.11}$	-	-2.36***	-3.02***	- / ** /***
234	-	-1.76***	-3.27***	-1.47*	-2.81***	-7.43***	- / - /***
1234	-	-	-2.24***	-	2.09***	-	- / ** / -

Table 26: Main effects and interactions for average Precision, Recall and FPR of experiment one and two for Ripper and all check-ins

Summarizing the results, experimental group one can be improved when using model 02, which results in an increase in average precision of 2.54 percentage points, along with an increase of average recall of 1.5 percentage points and an increase in average FPR of 9.11 percentage points. Note that these improvements are small relative to the ones achieved with Naive Bayes (7.45 percentage points average precision) and are even smaller considering the worse performance of Ripper's baseline model. For experiment two, no significant improvement can be achieved, thus the baseline model can be seen as the best model. Again, with Naive Bayes, an improvement of average precision of 13.69 percentage points was achieved for a baseline model that had better precision than Ripper's. Thus one can conclude that Ripper, even with additional feature groups, is inferior to Naive Bayes for evaluation with all check-ins.

6.4.1 Ripper and new Spots

Finally, Table 27 shows the results for the baseline of Ripper when only new models are considered.

Metric	Experiment 1	Experiment 2	Significance
Precision	0.037	0.046	***
Recall	0.071	0.110	***
FPR	0.034	0.045	-

Table 27: Average Precision, Recall and FPR for baseline models of Ripper with new check-ins for experiments one and two

Comparing the values with the results for Naive Bayes with new check-ins and Ripper with all check-ins, the changes of metrics between evaluations is different for Ripper than for Naive Bayes. For experiment

one, average precision drops to 55 percent points of the value with all check-ins, while this value is higher for experiment two with 68 percent. For Naive Bayes, precision lessened to 58 and 49 percentage points, which means that for experiment one and Ripper, the drop is more sever than for experiment two, resulting in a significant difference between the two experiments.
Average recall for evaluation of new spots with Ripper is also lower compared to the full evaluation, dropping to 46% (exp. 1) and 64% (exp. 2) of the values for full evaluation. Comparing to the differences of Naive Bayes, experiment one suffers much stronger percentage wise where the drop was only to 66% of the value for all check-ins.
Regarding average FPR, Naive Bayes with new check-ins has still a higher average FPR rate for both models, being 0.2 for experiment one and 0.17 for experiment two, while Ripper's average FPR is only 0.03 for experiment one and 0.04 for experiment two. This metric is the one where improvements relative to the full evaluation are present, with average FPR dropping to 64% and 83% of the full evaluation. Comparing the two experimental groups with each other, average precision now significantly differs between the experimental groups. For average recall, experimental group two still has a significant higher value while for average FPR, the difference is not significant.

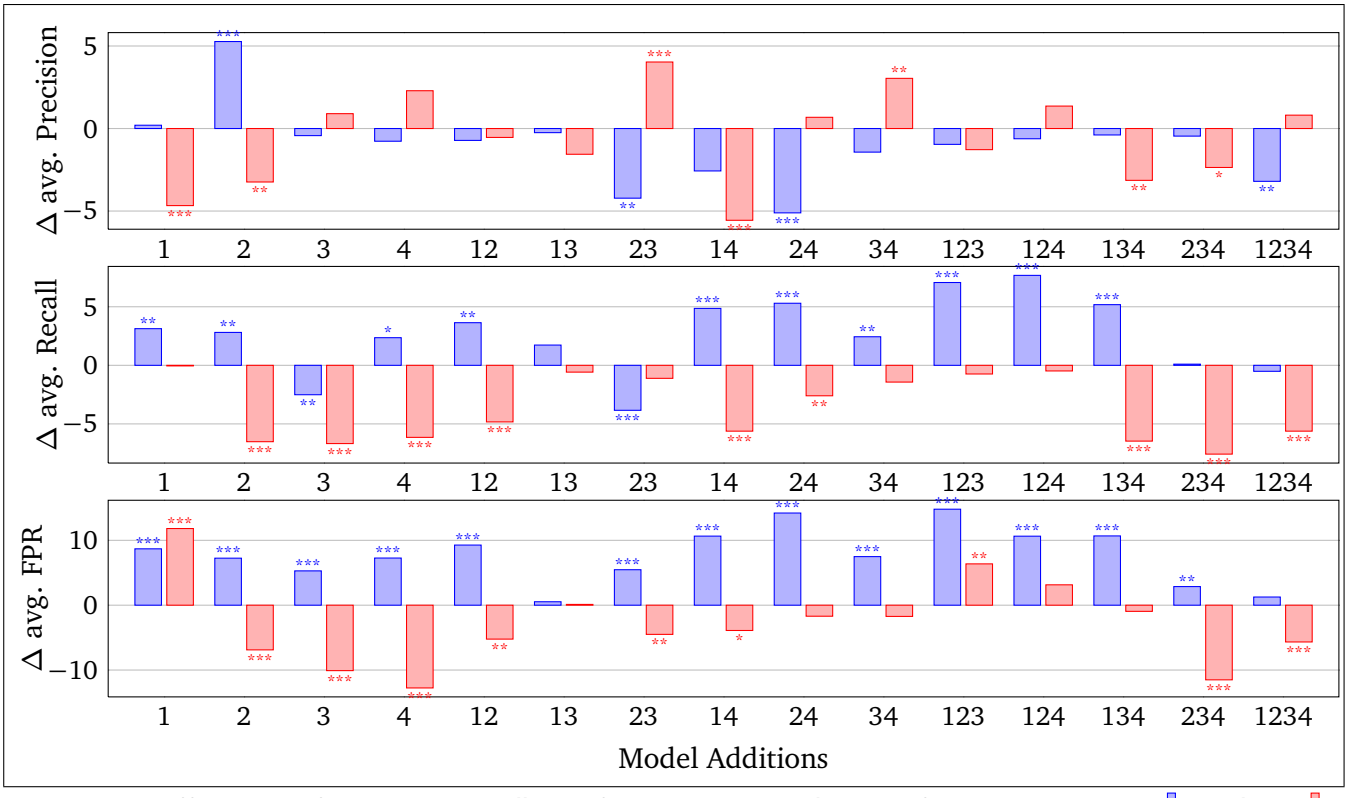


Figure 19: Differences of Precision, Recall, FPR for new spots and Ripper for experiment one and two

Changes in average precision, recall and FPR for different feature group combinations, as shown in Figure 19, show some differences compared to the evaluation with all spots. Again, model 02 of experiment one shows a significant improved average precision. But now, significant improvements are also recorded for models 023 and 034 of experiment two. For average recall, there is a difference in the general pattern between experiment one and two. For experiment one, most models show a significant increase in average recall, while for experiment two, most model show a decrease, with the majority of changes being significant.
Lastly analyzing average FPR, we again see differences between the experiments, as models from experiment one often show significant increase for all models, while for experiment two, the majority of models show a decreasing values, with the only exception being model 01, 0123, and 0134.

Taking a look at the main effects and interactions, we again see that interactions are present, though except for average FPR in experiment one, only one or two for each metric. Yet most of the main effects are not significant and generally of small magnitude. This prohibits to make general conclusions about the behavior of feature groups, as no significant generally impact of individual feature groups can be identified.

	Experiment 1			Experiment 2			Inter Exp.
Effect	Precision	Recall	FPR	Precision	Recall	FPR	Significance
1	-1.39$^{0.33}$	3.31***	1.90$^{0.16}$	-2.49 $^{0.11}$	0.96$^{0.37}$	6.68***	- / - / ***
2	0.30$^{0.83}$	0.52$^{0.64}$	1.86$^{0.17}$	1.02$^{0.51}$	-0.03$^{0.77}$	-1.04$^{0.48}$	- / - / -
3	-2.36*	2.73**	-2.53*	1.27$^{0.41}$	-0.50$^{0.64}$	-1.52$^{0.30}$	* / - / -
4	-0.34$^{0.81}$	-1.67$^{0.13}$	1.47$^{0.28}$	0.43$^{0.78}$	-1.93*	-3.25**	- /**/ **
12	-	-	-	2.80*	-	-	* / - / -
24	-	-	-3.75***	-	-	-	- / * / ***
34	-	-	-2.57*	-	-	-	- / - / -
123	-	-	2.70**	-	-	2.43*	- / - / -
124	-	-	-0.08$^{0.10}$	-	-	-	- / - / *
234	-	-1.97*	-3.39**	-	-3.84***	-8.33***	- /**/ ***
1234	-	-	-0.08$^{0.10}$	-	-	-	- /**/ -

Table 28: Main effects and interactions for average Precision, Recall and FPR of experiment one and two for Ripper and new check-ins

Thus, one can generally say that for experiment one, model 02 is the best with the only significant gain in average precision. For experiment two, model 023 showed the strongest improvements in average precision. In terms of average recall, model 0124 shows the best improvements for experiment one, and the baseline model remains the best for experiment two. For Average FPR, the best model for experiment one is 01234, whereas the best model for experiment two is 04.

6.4.2 Rule Statistics

Finally, we take a look at the rules generated by Ripper. Figure 21 describes the average number of antecedents originating from feature group one to four for each model. The remaining percentage is attributed to feature group zero.

For the single feature groups, the proportion of antecedents seems to remains constant over different models. For experiment one, friend features have the highest quota of antecedents with seven percent points, which is lower for experiment two with less than five percent points. On the other side, feature group four has for experiment two the highest quota of antecedents with nearly ten percent points, whereas this value is for experiment one much lower with 1.8 percentage points. Considering that for experiment two, more node data is available, Ripper seems to make use of these features more often.

Feature group two has the lowest quota of antecedents for both experiments with less than one percent point in both cases. This is remarkable, considering that for experiment one, model 02 was the best model. Thus this feature group seems to be used in only a few models, where it is an important antecedent for classification. On the other side, this seems not to be the case for feature group two.

Feature group three has also different quotas for both experiments, with the ones from experimental group two having on average 3.5 percentage points of their antecedents from this group while for experimental group one, this value is with 0.9 percentage points lower. As a reminder, for experiment two, it was indicated that the environment of features is richer. Like for feature group two, more data available seems to result in more antecedents originating from this feature group.

The model with the most antecedents for both experiments model 01234, where the models for experiment one have on average 10.2 of their antecedents from group one to four. This value is higher for experiment two, where 16.5 of the antecedents are from one of the additional feature groups.

The antecedents used from the different feature groups are often similar for both experiments. For feature group zero, the time of creation and the creator ID both form around 17 percent of all antecedents, followed by latitude, longitude, number of check-ins and number of visitors with 11 to 15 percentage points. This is interesting, as the two most widely used antecedents are usually not connected with the user's behavior. SpotID and name on the other side were only used for less than one percentage points. Words from the descriptions were in summary only used in two percent of all cases, with stop words being the most used ones.
For friend features, the feature indicating if a friend creating a spot were almost never used, with far less than one percent for both experiments. For feature group two, the feature describing the number of spots without a classification was used for 85% of all antecedents, with the next important feautre being the one describing the presence of spots from the category "Gas & Automotive" nearby. For feautre group three, restaurant and fastfood were the most important for experiment one with 12 percentage points, while for cafe, pub, railwaystation and bar all had around six percentage points. For experiment two, railway station was used for most antecedents with 13 percentage points, with restaurant, fastfood, cafe and pub each being used for six percentage points. Finally for feature group four, latitude and longitude were the most important features, being used for 60% (Exp. 1) and 47% (Exp. 2) of all feautres. Interestingly, latitude was used twenty percent more often than longitude. Following this, the URI of the node, different type descriptions like "Fast Food", "Railway Station" cafe and pub, were also common. From properties of the nodes, apart from geographical information, label and operator were most commonly used, with around one percent in both experiments. The number of rules and antecedents are

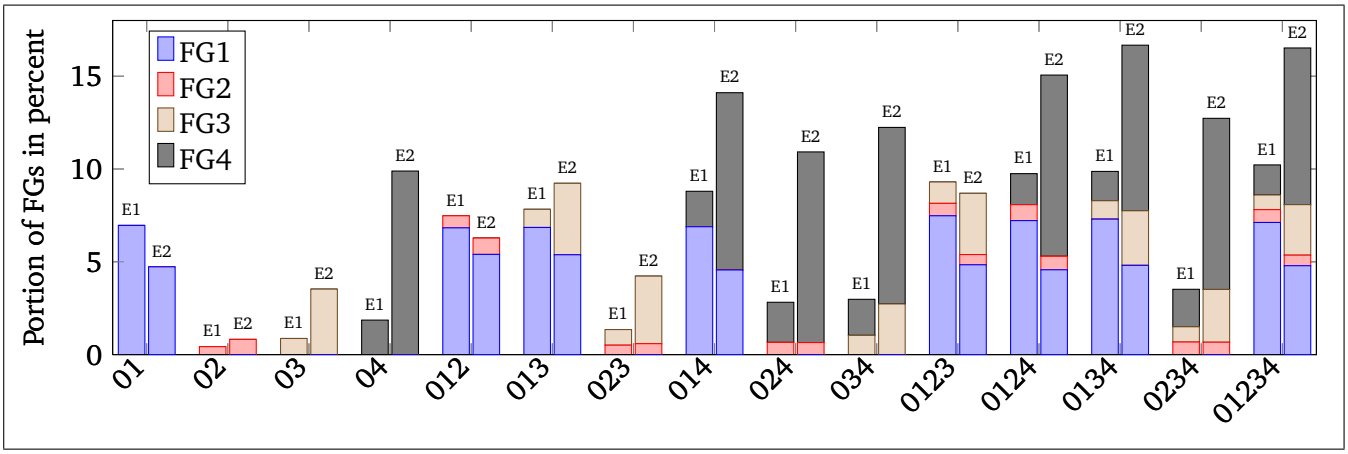


Figure 20: Average portion of antecedents from feature groups in percent for experimental group one (E1) and two (E2)

plotted in Figure 21. In general, Ripper produces only a few rules with each containing few antecedents. For every model, on average no more than six antecedents are used, which is a low number compared to the total number of available features. For both experiments, one can see that the number of rules tend to increase with the number of included feature groups. Models for experiment two usually have more rules and less antecedents than experiment one. Models from experiment one have on average the most rules for model 01, and the fewest rules for model 023. Models from experiment two on the other side the highest number of rules for model 0123 and the fewest rules for model 02.
Looking at the process of the rule graphs, one can observe that the differences between two models following after each other seem to be similar for both experiments. Exceptions are model 03 and 04, were experiment one declines while experiment two rises. Here, the change in average number of antecedents does not differ from experiment one, although the number of rules do. This leads to the conclusion that compared to experiment one the use of feature groups three and four fro experiment two leads to more rules, whereas the description length remains proportional.
For models 012, 013 and 014, there the changes are contrary, with models for experiment one showing an increase of the number of rules. Again, compared to experiment two, the different changes for the

number of rules is accompanied with different dynamics for the number of antecedents, which drops absolute as well as compared to experiment two.
Lastly, there is a hump in the average number of antecedents for experiment two which indicates that the description length is growing longer, for model 0123 in experiment one. Here it seems that with the combination of feature group one, with the highest number of rules and feature groups 23, which had in their model the fewest number of rules, results in an decrease in the number of antecedents, rather than a change of the number of rules.

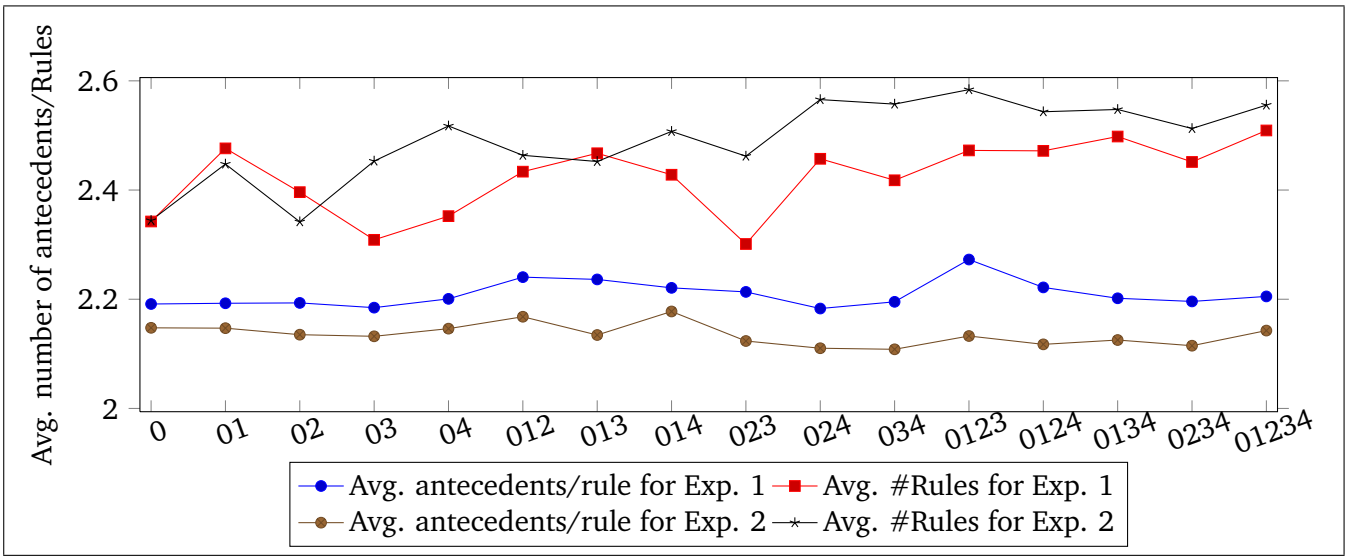


Figure 21: Average number of antecedents and rules per feature group for both models and experiments

In summary, Ripper is both inferior to Naive Bayes and also generally can barely make use of additional feature groups. If improvements are observed for individual models, they are much lower compared to the ones for Naive Bayes. As indicated by the antecedents from every feature group, Ripper seems to make use of the feature groups, also proportionally to the data available for each one. On the other side, Ripper generates relatively few and short rules, which are maybe unsuitable for the classification task at hand. Thus the general attempt of creating short rules might be a hindrance for content-based classification of POIs, especially as the feature groups introduced consists of many different features with often missing values.

6.5 Learn Times

Finally, one aspect to shine a light on are the learn times for the two classifiers and different models, as shown in Figure 22.

As can be seen, Ripper generally has learn times around one minute with a minimum of 58 seconds

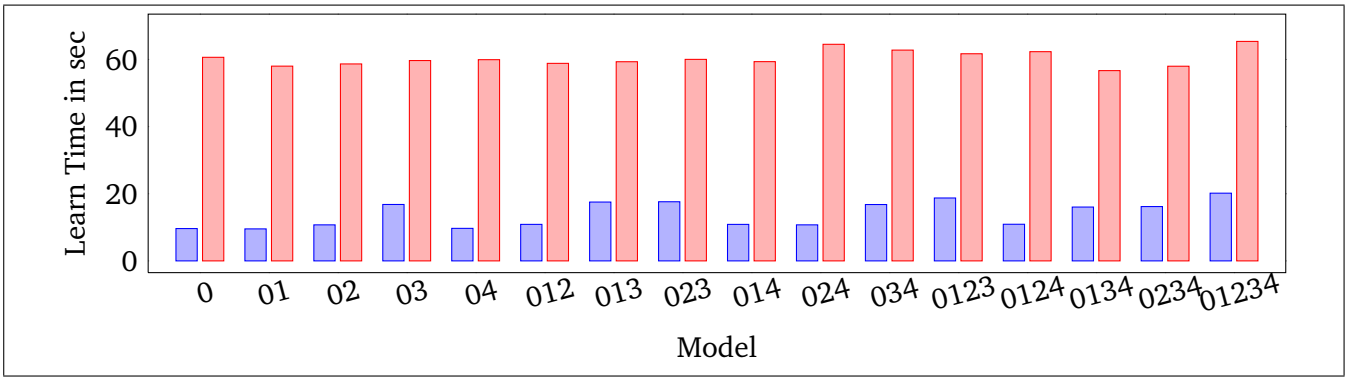


Figure 22: Learn Times for Naive Bayes and Ripper on different Models

for model 01 and a maximum of 65 seconds for model 01234. Naive Bayes on the other side has much shorter learn times, with a minimum of 9.5 seconds for model 01 and a maximum of 20 seconds for model 01234. For Naive Bayes, learn times are relative long for models where feature group three is involved. This may be due to the fact that this feature group has the largest number features, making it time-consuming to compute statistics for each of them. Still, Naive Bayes generally has much shorter learn times, at the worst case taking on average 23% of Rippers learn time. This is another advantage of Naive Bayes, besides its better results, the higher ability to cope with additional feautre groups and the absence of interactions.

6.6 Context Recommendation

In this section, we discuss the results when contextual post-filtering is used in conjunction with content-based recommender systems. The following sections report the results for the two types of context we used, weather and time.
As we are more interested in the general impact of context-based recommendation, we pooled the two experiments for finding the optimal threshold, rather than optimizing it for every feature group separately. Also, we will concentrate more on the general impact of our approach. Because of this, we will regard post-filtering as an additional factor in our factorial design, deferring individual models and concentrating on the main effect.

6.6.1 Weather Context

When optimizing the thresholds for post-filtering using weather data, it turned out that the optimal values for all models and both classifiers where the ones that lead to no adjustments by context, i.e. both a downgrade and upgrade threshold of 1.0. This means that including the context of weather actually led to a decline in average precision. As an example, Figure 23 displays the change in average precision relative to the thresholds for the baseline model of Naive Bayes, summarizing both experiments. As can be seen, the fare right corner represents the optimal solution and every threshold less than the maximum one leads to negative changes in average precision. Also note that the worst possible change is very small, being just -0.2 percentage points of the original value.

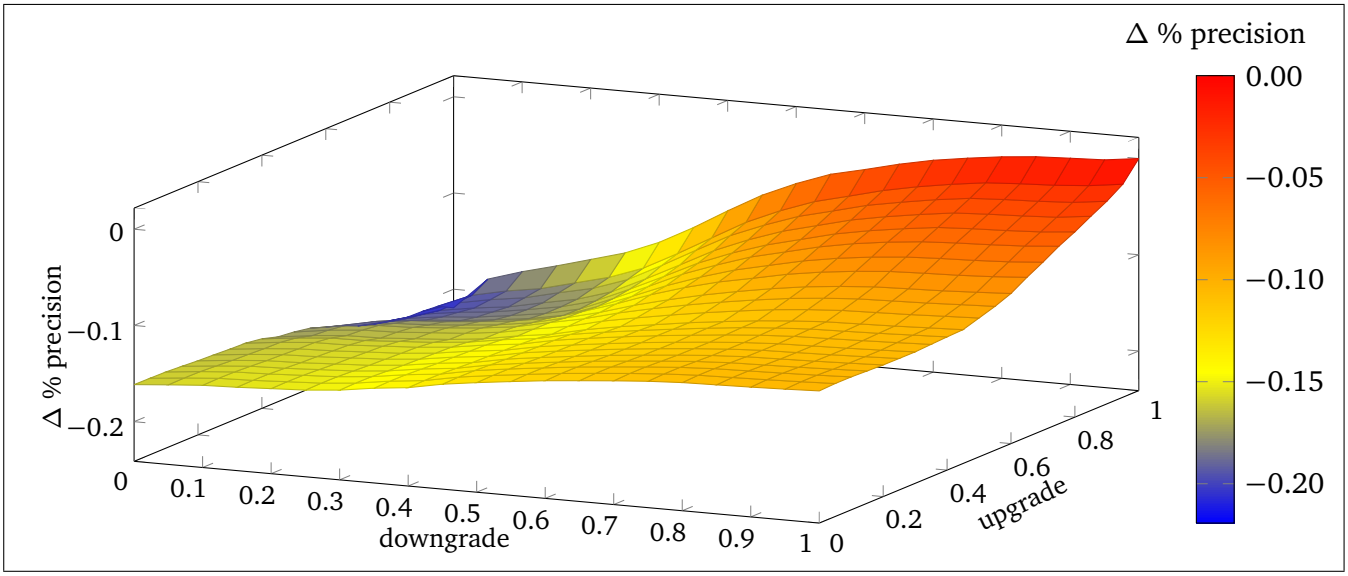


Figure 23: Precision in dependence of downgrade and upgrade threshold

When searching for the reasons of this behavior, two possibilities arise. First, our approach of post-filtering might not work at all. As we will see in the following section, it works when using time as a

context, which brings us to the second possible reason, namely that the representation of the weather context we chose is not appropriate.
Indeed, a plausible possibility might be that weather as context is more complex than anticipated. In our approach, we put together data from different climate zones, which alone are quite different from each other. Taking the example of the U.S. east coast, which in winter is frequently struck by snow blizzards and Germany during the same time, with a comparatively stable climate, it becomes obvious that even within climate zones, great variations exist. Thus our approach of putting all these values together seems to have created models that are not able to contribute substantially to forecast uses's behavior. For further studies, more detailed models, possibly with a location-based segmentation, should be used in order to create models for climatic homogeneous areas.

6.6.2 Time Context

After weather context, this section discusses the results for the use of time context for adjusting recommendations given by our content-based recommender system. Compared to weather context, the addition of time leads to an significant increase in average precision. When determining optimal thresholds, the values are surprisingly similar for different models from the same classifier. Models from Naive Bayes have an optimal downgrade threshold between 0.73 and 0.75 while the optimal upgrade threshold lies between 0.89 and 0.90. For Ripper, the optimal downgrade threshold is between 0.76 and 0.77, and the optimal upgrade threshold for all models is 0.66.

Using the average of the thresholds of the models from one classifier as the threshold for every model and evaluating each one with all check-ins, average precision for models from Naive Bayes increased between 0.87 and 1.20 percentage points, although only the changes for models from experiment two are significant. Average recall drops significantly between 1.50 and 2.12 percentage points, while average FPR improves between -4.36 and -5.58 percentage points, also significant for every model.
For models generated with Ripper, even stronger significant improvement are reported. For average precision, the increases range from 9.64 up to 15.19 percentage points. For average recall, the improvements range between 73.30 and 83.06 percent points. At the same time, average FPR is also raising, ranging from 83.92 percent points up to 99.07, nearly doubling the value for FPR.

An analysis show that no significant interactions between the use of post-filtering as a fifth factor and other feature groups are present, we report its main effects in Table 29, describing the general impact. Again, the improvements are to be seen relative to the baseline model of every experiment.
As already shown, the improvements achieved with context-based recommendation used with Ripper is much stronger than with Naive Bayes. For the latter, we see an significant improvement in average precision for the first experiment of 1.32 percentage points, accompanied with an decreasing recall of 1.89 percentage points and an improving average FPR of 5.04 percentage points. For experiment two, there is only a minor and insignificant improvement for average precision, along with a drop in average recall and average FPR.
For Ripper on the other side, the main effects on average precision are 13.48 percentage points for experiment one and 7.01 points for experiment two. Average recall also strongly, adding for both experiments roughly two thirds of the baseline value. But average FPR grows too and even stronger, with an decrease of around ninety percent for both experiments.

With these strong improvements, the gap in terms of metrics difference between Ripper and Naive Bayes has nearly closed, at least for experiment one. Here, the best model for Ripper (02 with context recommendation) has an average precision of 0.077, compared to average precision of 0.079 for the worst model for Naive Bayes, model 0 without context recommendation.[121] Also there are significant different

[121] With a formal non-significant difference, although the p-value of 0.109 is very close

Classifier	Experiment	precision	recall	FPR
NaiveBayes	One	1.32***	-1.89***	-5.04***
	Two	$0.62^{0.25}$	-1.93***	-3.69***
Ripper	One	13.48***	72.17***	89.40***
	Two	7.01***	76.29***	93.96***

Table 29: Main Effects on precision, recall and FPR for both classifiers and experiments when evaluated with all check-ins

values for recall (0.27 for Ripper, 0.37 for Naive Bayes) and average FPR (0.1 for Ripper, 0.2 for Bayes). For experiment two, the gap is wider. The best model for Ripper, using only baseline features with context-recommendation, has a average precision of 0.071, recall of 0.298 and FPR of 0.103. Compared to this, the worst model for naive Naive Bayes, model 01 without context recommendation, has significant higher values, with an average precision of 0.1, recall of 0.45 and FPR of 0.14.

Only considering new spots, we see the main effects in Table 30. For Naive Bayes, changes of metrics are smaller for average precision and recall, but not for average FPR. For experiment one, increases in average precision is smaller, which is also true for the decrease in average recall. Average FPR on the other side has still a decrease of around five percentage points.
For experiment two, precision is not changing, and the change in average recall is not significant. Average FPR on the other side is decreasing with 5.51 percent points stronger than for all check-ins.

Classifier	Experiment	precision	recall	FPR
NaiveBayes	One	1.09*	-1.45***	-5.21***
	Two	$0.01^{0.99}$	$-1.79^{0.70}$	-5.51**
Ripper	One	47.9***	143.3***	136.14***
	Two	17.93***	92.75***	103.83***

Table 30: Main Effects on precision, recall and FPR for both classifiers and experiments when evaluated with new check-ins

For Ripper on the other side, the changes in average precision increases rapidly, with nearly fifty percentage points for experiment one and seventeen percentage points for experiment two. The changes in average recall doubled for experiment one and increased moderately for experiment two, which can also be said about the increases of average FPR. As it seems, Ripper profits especially for new spots from the use of contextual post-filtering.
Using this approach, Ripper is not only able to return much better results, but becomes competitive to Naive Bayes for recommendation of new spots. For the latter and experiment one, the best model uses feature groups one and four, along with context-recommendation, and yields an average precision of 0.048, an average recall of 0.25 and an average FPR of 0.20. With the best model of Ripper (024 with context recommendation), average precision is significantly higher with 0.56, while average precision with 0.18 and average FPR (0.09) are significantly lower.
For experiment two, Ripper coupled with a time-based, post-filtering extension not only surpasses the worst model of Naive Ripper, but even its best. When using Ripper in its best configuration (model 034 with post-filtering), one gets a precision of 0.055, a recall of 0.211 and an FPR of 0.09. Contrary to this, the best model for Naive Bayes, model 0234, has a average precision of 0.042, an average recall of 0.16 and an average FPR of 0.1.
In summary, this means that when using post-filtering, even a weak classifier like Ripper can become competitive to an well-performing recommender.

7 Conclusion

In this thesis, we enhanced POI recommendation for a location-based social network by enriching it with semantic data. Specifically, we evaluated the two hypotheses that (1) the addition of knowledge to the existing location data enhances recommendation and that (2) the addition of knowledge to the existing check-in data enhances recommendation. As a basis, we used a crawled data set of Gowalla, a location-based social network, where users could publish their position and share it with other users by checking-in at POIs called spots.
To evaluate these hypotheses, the first task was to identify and integrate additional knowledge, for which we used two sources. The first one, LinkedGeoData, a project that offers linked data of POIs, served for enhancing knowledge about Gowalla's spots. The second one, MesoWest, a source of historical weather data for North America, enabled us to add new knowledge about check-ins.
To automatically link nodes from LinkedGeoData to spots in Gowalla, we first matched them through a custom heuristic, comparing names, positions and classifications, which resulted in finding matches for 11% of all spots. Then, we used FeGeLOD, a tool for unsupervised creation of feautres from linked data, to generate additional features for spots, based on the linked nodes' attributes. Also using LinkedGeoData, we created an additional set of feautres, describing a spot's environment, using the types of nodes that were in the vicinity of a spot.
To make more use of the available Gowalla dataset, we created two additional feature groups, one being a group of hybrid, friend-based collaborative features, describing how affiliated a user's friends are with a spot, and a description of a spot's environment through categories of spots in its vicinity.
This resulted in a total of four additional feature groups, which could be added to the baseline feature group, generated from the available profile information of a spot.
For acquiring supplementary data about the weather conditions under which a check-in occurred, we implemented a custom crawler for MesoWest, using it to download weather information for roughly half of all check-ins. As additional contextual data, we used the timestamps of check-ins to make use of temporal patterns emerging from check-ins.
For our experiment, we implemented a content-based recommendation approach with two different classifiers, Naive Bayes and Ripper. We sampled two groups of one thousand active users, with one representing the average user and one representing users who often visit spots with matched nodes. For every user, we learned preference models for all sixteen feature group combinations using both classifiers, resulting in 32 models per user. Learning was done by presenting the checked-in spot as a positive example and spots in the surrounding of the checked-in spots as negative example to the classifier, simulating a user's process of considering the spots in his surrounding prior to performing a check-in. The same approach was used for evaluation of every check-in, were a classifier rated the visited spot and those in its surrounding either as visited or not visited. For evaluation, we primary relied on average precision, which was suitable as central metric in our setup, as only one item could be categorized as true positive.
For including time and weather contexts in our process, we added a context-aware, post-weighting extension for our recommender. For this, we learned models for both time and weather, describing the probability that a user checks-in at a category of spots, given the predominating weather or time. Using these models, we adjusted the list of recommendations, given by the content-based recommender, depending on the plausibility of the classifications, given the predominant context situation.

The results of our experiments confirm the first hypothesis. Through the use of additional knowledge about spots, especially features created from LGD data, we were able to enhance prediction accuracy for both experiments. It showed itself that the two classifiers used are differently fit for such a task.
When using Naive Bayes, using additional feature groups generally enhanced average precision. We were able to enhance precision up to nearly fourteen percent points, along with higher average recall of 12.29% and a decrease in average FPR of 3.41%. When only considering spots the user had not visited

before, we were able to increase average a precision up to 13.92 percentage points, along with a higher average recall of 10.83% and an increase in average FPR of 9.29%, although this was only achieved by ommitting environmental descriptions, as they harmed average precision.
The feature group based on additional node data, which was the core of our setup, improved recommendation in all cases. We could also show that the gains were significantly higher if more spots with node data were available.
Evaluating our setup as a factorial design, we showed further that the effects of feature groups were usually significant and mostly showed no interactions between each other, which proves that the impact of feature groups can be seen independent from each other. This is not surprising, as the independence assumption used for the Naive Bayes classifier makes the contribution of feature groups to the classification of a spot independent. Interactions between feature groups were only noticeable if the effect of one feature group was particularly strong, in which case it interacted with other groups at a comparatively small magnitude. Thus for Naive Bayes, an improved scheme for testing whether new feature groups should be used or not with different feature group combination can be proposed, were the effects of an individual feature group is first assessed isolated. If the impact is positive, one can add it to other models, analyzing potential interactions between feature groups. This could save a lot of effort in a setup like ours, as not every combination of feature group needs to be evaluated.
Seeing that the inclusion of feature groups is not appropriate for every situation, Naive Bayes can further be modified in a way to create one model with different features or feature groups that can flexibly used or not for a classification task, depending on the user and/or the situation. As the computation of a class probability is based on the multiplication of the feature's probabilities, one could dynamically specify which features should be included or not, depending on the situation. For example as we have shown that for new spots environmental factors are not beneficial for recommendation, one could exclude them if a user visits an area he has not visited before. This would make it unnecessary to create special models for this case, reducing the model learning effort for users.

Ripper on the other side already had inferior performance if the models only using baseline features were compared to the ones created with Naive Bayes. Using additional feature groups increased average precision only up to 2.52 percentage points for one model, with this improvement being only achieved if data from LGD was excluded. When considering only spots the user did not visit before, improvements up to 5.27 were recorded, this time also with features from LGD. These small improvements failed to make models learned with Ripper competitive to those created with Naive Bayes.
Also, the effects of feature groups were often of small magnitude and not significant, indicating a big variability, and exhibited manifold interactions between each other, which is why no general impact of feature groups on model performance could be shown. These strong interactions can be attributed to the growing phase of Ripper, where the decision which antecedent to be added next depends on the antecedents and rules generated previously, making the use of features from feature groups and thus their effects contingent upon on another.
The reason for Rippers inferior results might lie in its effort to keep the number of rules and antecedents low, leading to underfitted models. Additionally considering the learn times of Ripper, which were on average up to three times longer than those from Naive Bayes, this classifier and probably rule learners in general seem to be inappropriate for spot recommendation, especially with more features added.

The second hypothesis, stating that adding knowledge about check-ins enhances recommendation, could not be confirmed. When using additional weather data for context-aware recommendation, average precision commonly dropped. In our opinion, this should not be contributed to the additional data itself, but to the way we made use of it. In our opinion, the models we created for determining the probability of a check-in under a given weather context where unsuitable, as they pooled data from different climate zones into the models. This created models without explanatory power, resulting in no increasing recommendation performance.

Obviously, there are possibilities on how to make better use of the contextual data. First, one should segment it to get models for climatic homogeneous areas. This could be either done with a heuristic, for example simply taking tiles of 25 square kilometers as one unit, through the use of climate models or automatically, by beginning with small models and joining those which show a high resemblance. Then, one could either use these segments to compute the deviations from the models average for every check-in, pooling these values into different models for spot categories, or creating models for each segment individually. With such refined models, we think that it is possible to improve recommendation results through the addition of weather data.

We also performed context-aware recommendation using the context of time, where we could show that our approach of post-weighting actually can improve recommendation. But as this data is not truly additional knowledge, we did not use the results as the basis for evaluating the hypothesis. Using this approach, we were able to further enhance recommendation for models using Naive Bayes significantly to a relative small degree. For Ripper on the other side, the improvements where much stronger, up to the point where these models became comparatively with Naive Bayes regarding performance. When considering new spots only, Ripper in conjunction with context-based filtering even surpassed the results for Naive Bayes for one experimental group, resulting in better precision and recall, along with a lower FPR.
This last finding came at a surprise and was not expected. But as Ripper previously had worse results than Naive Bayes, and especially a low average FPR, we think the ratings made by Ripper gave the context adjustment mechanism the ability to recommend many spots, leading to increased metric values. Naive Bayes on the other side had already a comparatively sophisticated classification for spots, making it more difficult to find a threshold that could positively influence the recommendation performance. In our opinion, a solution to let Naive Bayes profit more from context-aware recommendation could lie in pre-filtering spots depending on their appropriateness in the given context and then presenting the reduced candidate list to Naive Bayes. This would make the addition of contextual knowledge independent from the preconditions inflicted by the classifier, giving more freedom to include it.

Our approach leaves some questions regarding the impact of additional knowledge unanswered. While we concentrated on users with more than fifty check-ins in order to get expressive user models, it would be interesting to take a look at the cold start problem, for example by creating models for our users every ten check-ins and using these models to evaluate the remaining ones. This would show at which number of training check-ins different feature combinations show improvements. It could clarify if additional feature groups can maybe mitigate the cold-start problem, as their inclusion may lead to a rise in recommender performance prior to the point were this is the case for models using baseline features alone.
Also, additional classifiers may be considered, for example Support Vector Machine (SVM) . While we initially tried to use this classifier, we failed to learn models for users with larger numbers of features and check-ins at acceptable times, which might be due to the fact that the high number of points and dimensions were hard to handle for a SVM.
Taking a look at our experimental design, some improvements can be applied for more realistic results. With our approach, we oriented ourself on real world recommender of LSNs, were a user gets recommendation with respect to his current position. Especially the concept of area could be refined to better resemble his considerations for training data as well as test data, where recommendations are given for an area. An alternative to our approach of using air line distance to define a list of candidates could be to use the nearest fifty spots around a checked-in spot or to chose the middle ground, for example by taking all spots one kilometer away from the checked-in spot and if there are less than fifty spots in this range, to take the fifty nearest. This might be a compromise between recommending spots in a city, where there a plenty of possibilities within walking distance of one kilometer, and recommending spots in more suburban or rural areas, where spots have a higher distance between each other, but where it is also normal to drive greater distances to get somewhere. Finally, areas defined on actual traveling distances

would be a more elaborate design, where spots nearby are identified by their distance in terms of driving distance on roads and public transport. For this, suitable data about road systems and public transport needs to be available, which at the moment can already be found as linked data in LinkedGeoData.
For context-based recommendation, the segmentation of models used for evaluating the plausibility of a check-in given a context could be improved. This could be done based on any properties of users (age of users), spots (location of spots) or even check-ins (two models for weekdays and weekend days for the probability that a person will check-in in the evening at a bar). Such models could be used for pre- and post-filtering, in the first case removing implausible check-in candidates both from training and test check-ins, in the latter case re-classifying spots that have been classified implausibly. Pre-filtering could also be done to created different user models, for example for a weekday or a weekend day or a warm and a cold day, learning models on check-ins that happened in the specified context segment and using the corresponding model if a recommendation should be done given a specific context. If in the future, more approaches for contextual modeling show up, these could be applied too.

Taking a look at the different feature groups, there are also some ideas for possible improvements and approaches for acquiring useful features.
Feature group one, containing friend features, showed a mostly positive impact, especially for spots not seen before, which confirms that friends play an important role for recommending spots to users, an insight that is consistent with previous work. It seemed though that with fewer friends, Naive Bayes tended to recommend too many spots, resulting in a harmed average precision. Ripper on the other side was not able to make use of these features, although a considerable number of antecedents came from this feature group. A refined use of friend connections could be to see the location of the user's friends as context information related to a check-in. With friends nearby, a user might be inclined to go to spots where his friends are, while if none of his friends are around, he is more affected by his personal preferences. As only the arriving of a person at a spot is recorded, an open question would be how to determine if a friend is at a spot. Without additional knowledge, one has to resort to approximations when defining when a user is at a spot and when he left.
A additional feature group related to the problem of actual presence of a user can be to create a contextual dimension of the places a user has visited before. There has been some research on this topic where it was shown that transition patterns of users between categories of spots exist.[122] Again, a problem here arises to determine when a user actually performed a transition from one place to another and when he visited other places in between.

Regarding the two feature groups describing a spot's environment, one could see that such groups led to decreases in average recall and FPR which was stronger if the user had not visited the spot before, resulting in a lower average precision. This seems to indicate that these feature groups were not able to generalize spot environments well,leading to overfitted models. For Ripper, especially feature group two was part of models with improved precision, although the feature group's impact was not consistent for different feature combinations.
An idea to avoid overfitting with these feature groups could be to describe the environment as quotas of different types present in the environment. This would make it irrelevant if the environment consists of five cafés and five shops or ten cafés and ten shops. Alternatively, some dimension reduction approach like PCA could be applied, which could reduce the environment to more abstract concepts, like "work environment" "leisure time environment" or "shopping environment". Especially for the environment created with LGD nodes, which had more than 5000 features, this approach might enhance generalization.

Finally, feature group four, which contained features derived from matched nodes, showed a positive impact if it was integrated into a model that was trained with Naive Bayes. The difference between the

[122] For example from airport to train station, see Noulas et al. [44]

experiments showed that having more matches between spots and nodes can enhance recommendation further. For Ripper on the other side, it showed improvements only for one particular feature group combination.
As described in Section 4.2, the availability of matches was with 11% fairly low. This was not unexpected and can be contributed to the conceptual differences between the two systems and their different POI distributions, as outlined in Section 3.2.1 and 4.1. Judging from our experience with the two datasets, we think that the quota of matches is pretty close to the maximum possible. Yet maybe some more matches could be found if only association rules with a minimum level of support would be used for matching nodes by category. Because if an item support is too low, a lot of rules with a low rule confidence are generated, possibly creating a lot of inexpressive rules.
Regarding importing data with FeGeLOD, some additional filters could be created. For example, one filter could do the "additional hop", creating features based on entities that are connected through another entity to the one for which features should be created. Taking LinkedGeoData, if the street properties of nodes were no literals, but linking to an entity representing the street, such a strategy could describe the types of nodes that are in the same street like the entity we are interested in. Certainly, this can easily result in a large number of additional features, for example if the linking node was the state a node is located in. One way to mitigate this problem could be to restrict oneself to "bridging entities"[123] that have not too many other entities linked to with the same relation, as one could suggest a rather special connection between them over the bridging entity.

But for such additional strategies being fruitful, in our opinion LGD itself, more specifically the interlinking of its entities, has to be improved. A typical node in LinkedGeoData is mainly linked with its types, its shape, the last editing user and sometimes, external resources, while for example its address is a set of literals.[124] By applying standard record linking methods, a node could be linked to entities he is related to, like its neighbors, its street, its neighborhood, its town or its state, densifying the graph of nodes and their connections and leveraging the creation of additional features, as more relations to other entities can form the basis for feature creation strategies.

As we have shown in Section 3.2, the known LOD cloud currently does not contain more resources that are suitable to be linked to the Gowalla dataset. But taking a look beyond this index, there are other sources which can serve as a potential source for features.
If one searches for other collaborative mapping projects similar to OpenStreetMap, WikiMapia[125] proves an alternative. Compared to OSM, its seems to be more focused on spatial areas, such as villages or quarters, rather than individual POIs on a global scale. This would constitute a nice supplement to the existing data from OSM, as data on small-scale areas, like quarters, might be helpful to see a user's preferences for them, for example if he likes an amusement quarter or tends to avoid it. As the data is not published as linked data, tools like FeGeLOD could not be used, unless the data was first transformed into linked data.

For the future, a prime source for geospatial linked data might be data published by governments, like in `data.gov` for the U.S. or `Data.gov.uk` for the United Kingdom and possibly further down the road, similar German and European projects. As public administration collects manifold data from many domains, such projects would provide an abundance of data about spatial objects. Examples are, apart from cartographic data, statistic and information about different administrative regions (environmental protection areas, industrial areas), transport (road network) and location-dependent sociological information (e.g. criminal statistics). At the moment, only very few linked data is available from `data.gov` and `data.gov.uk`, and even less has a geospatial component. With more data becoming available, and the publishing of linked data being more common, such projects can also become a prime source for

[123] i.e. entities over which a connection between to entities exist
[124] See the example of a Node in Section 3.2.1
[125] `wikimapia.org`

additional features for recommendation. For example, if the results of food inspections where published, pre-filtering could be done for badly rated restaurants, which would constitute an additional service to the user.[126]

Taking one step back, bringing spatial entities in relation to each other can in our opinion be very beneficiary, not only for discovering new information like we did with the environment description for spots, but also for many other applications. But the handling of geospatial objects and their relations need special considerations beyond simply following links between the entities.
In general, spatial relations between entities can be either qualitatively, for example if one entity lies within the other or shares a border with each other, or quantitatively, for example by expressing the distance between to entities.
Qualitative relations are covered by the DE-9IM standard, which defines spatial relations between two geometries in the two-dimensional space. As these relations are binary, they can easily be represented as linked data. with respect to these relations, ambiguity can only arise from the data itself (if different data sets indicate that different relations between two entities exist), not the relations itself. Also, for querying such relations, there has been standardization efforts of the OCG by introducing the GeoSPARQL standard, which defines core vocabularies for describing basic geometrical objects, as well extensions to SPARQL for allowing more complex spatial queries covering these relations.[127]
Quantitative relations on the other side are more complex, as they are ambiguous. Depending on the application, for example the distance between to objects can be defined differently (distance between two towns as distance between city limits or between city centers) and a categorization like “near”/”far away” highly depends on the application (if you have a helicopter, Cologne is near Frankfurt while by foot, it is far away).
The ambiguity of such relations make it hard to interpret such quantitative statements[128], unless there is an ontology with which such relations can be explained. Although advances in the research field of geospatial semantics will likely lead to such an ontology, in the near future such relations needs to be defined depending on the application due to a lack of such a vocabulary.
Thus if one wants to have the ability to create features based on entities that are in a qualitative or quantitative relation to the initial entity, one has to create a domain specific extension for geospatial objects. A wide implementation of GeoSPARQL in triple stores can ease the task of finding such relations. Alternatively, explicitly creating statements describing such relations are possible, though here too, no ontology currently seems to be available.
Quantitative relations on the other side have to be inferred by the intelligent agent. For this, it would be beneficiary if different geospatial data sources would use the same vocabulary for describing basic geographic entities like points, lines and polygons.[129] Although such vocabularies exist, for example the NeoGeo vocabulary[130] or the GeoOWL vocabulary[131], there seems to be no de-facto standard and it remains to be seen if such a standard will emerge in the near future.
But assuming such a standard in place, a tool like FeGeLOD could be extended with specific rules for geospatial data. By using qualitative relations, possibly using GeoSPARQL queries and by defining distance measures between different entities and “nearness” definitions, one could relate entities to each other, for example not only acquiring the types of the nearest spots, but also creating features indicating if the spot is next to a highway, near a lake or with an amusement park nearby. Standardized vocabulary for geometrical objects would ensure that such filters are kept generic, allowing them to be used with different data sources.

[126] This would then be a rule-based recommendation system, as it is not based on user’s preferences
[127] see `http://www.opengeospatial.org/standards/geosparql` for details
[128] For example in the geonames ontology, there is a predicate “nearby” where it is unclear what it actually means
[129] The only widely used W3C geo vocabulary restricts itself to describing point
[130] `http://geovocab.org/doc/neogeo.html`
[131] `http://www.w3.org/2005/Incubator/geo/XGR-geo/W3C_XGR_Geo_files/geo_2007.owl`

Inferring on geospatial data this way would not only give rise to additional possibilities for feature creation, but could be used for every application where the spatial positions of objects count. If one would for example search for a hotel near the coast in south France, an intelligent agent would be able to search in different databases for hotels that fulfill some predefined condition of nearness, relieving the user from the task of manually evaluate the position of Hotels or to rely on some pre-defined categorization from hotel databases. It would also enable to perform reasoning for more complex tasks. For example, having the ability to uniformly access spatial information from different sources and perform spatial reasoning could lead to potent tools for location planning at different levels. For example one could search locations for a new factory by entering desirable criteria, such as the requirement for a highway and a river nearby (for transportation), with no railways 2 kilometers around (because of the possible vibrations affecting the high-precision machines) in an administrative area with low taxes and a high quota of sun hours (for the solar energy plant that should be installed as part of the corporate's environmental responsibility program). An agent could join information from different sources (for example LinkedGeoData for roads, rivers and railways, open government data on taxes and databases from NGOs on solar radiation values for different areas) and perform reasoning on them, finding a list of potential locations. One step further, a chain could even use a content-based recommender system for location planning, classifying its chains with a desirable factor (like revenue) and create features from all kinds of spatial databases. With a learned model, such a system could estimate revenues for potential sites and thus give recommendations on where to place the next stores.
In summary, adequate vocabularies and tools for handling geospatial linked data, can enable intelligent agent to uniformly access and integrate very different kinds of spatial information, leading to new possible applications in this area. Given the early stage in this development, we assume that it will take some more time and research effort to make these ideas feasable.

A Appendix

A.1 Bezier Curve

The quadradic Bezier Curve is then defined as:

$$C(t) = \sum_{i=0}^{2} \binom{2}{i} t^i (1-t)^{2-i} P_i \tag{28}$$

$$= (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2 \tag{29}$$

$$= (P_0 - 2P_1 + P_2)t^2 + (-2P_0 + 2P_1)t + P_0,\ t \in [0,1] \tag{30}$$

This function can be split into two separate quadratic functions for the x and the y-value of a given point c on the curve. Additionally, as the coordinates of the points P_0 and $P_2$are known, they can be inserted into this quadratic functions. This leads to:

$$c_x(t) = (-2x_1 + 0.3)t^2 + 2x_1 t,\ t \in [0,1] \tag{31}$$

$$c_y(t) = (-2y_1 + 1)t^2 + (-2 + 2y_1)t + 1,\ t \in [0,1] \tag{32}$$

where $P_1 = (x_1, y_1)$ the second control point and the parameter for this curve.
From these two curves, one can determine for a given distance a value in $[0,1]$, depending on the Bezier Curve. This works by first solving the quadratic equation $c_x(t)$ for t. If it happens that the term $-2x1+0.3$ happens to be zero, this means that the remaining linear equation will be solved. Otherwise, the two roots will be calculated and the solution within the boundaries of $[0,1]$ is picked. Then, this solution is used to calculate $c_y(t)$, giving the desired value.

A.2 HeatMaps

In the following, different heatmaps are presented. The first two show the density of spots and of check-ins on the world map. The third shows the density of nodes from LinkedGeoData on the world map. The numbers where obtained by counting the number of nodes within a degree of latitude and longitude.
The last figure shows the density of matches between Gowalla spots and lgd nodes.

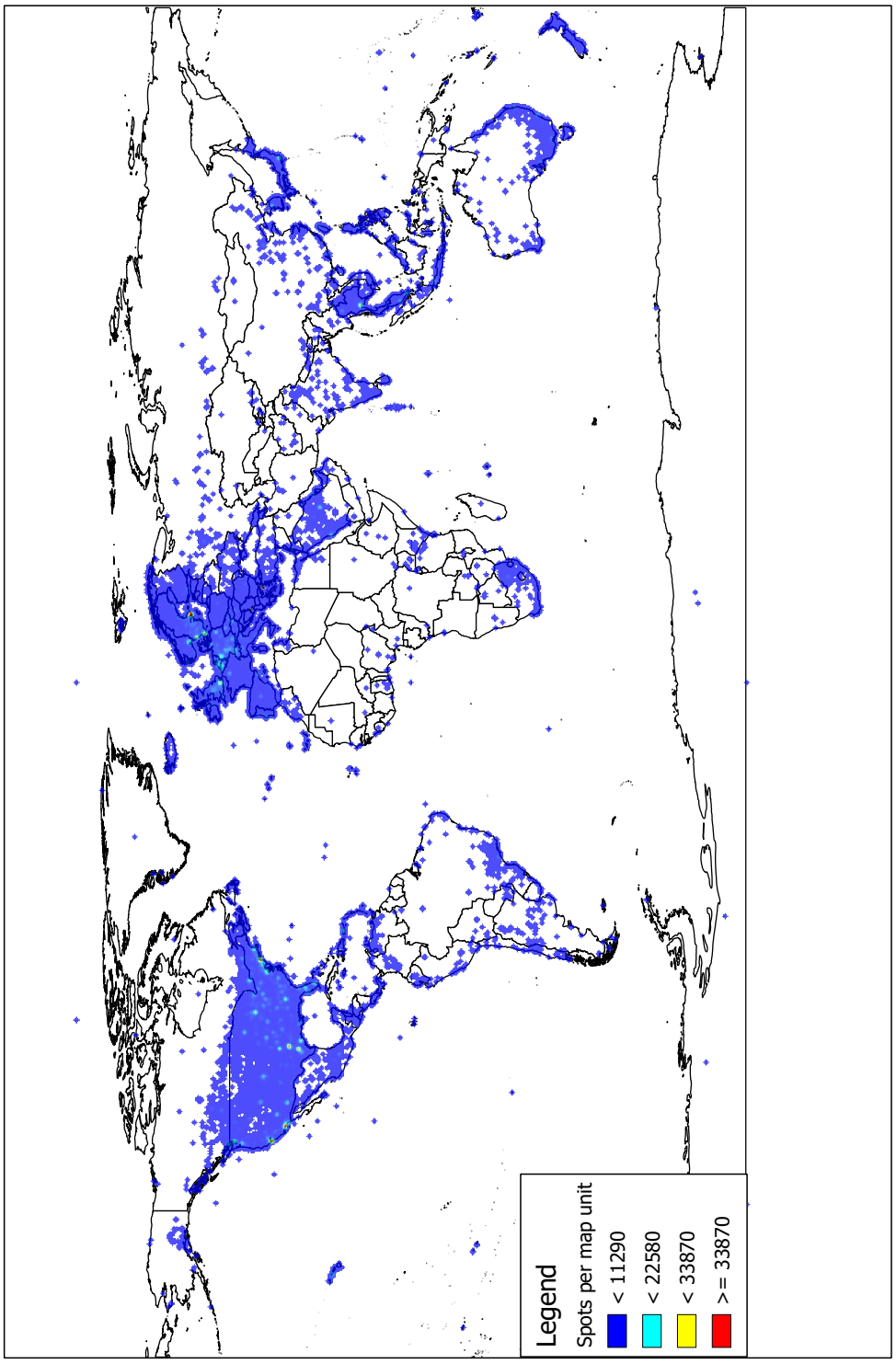


Figure 24: HeatMap of Gowalla Spots

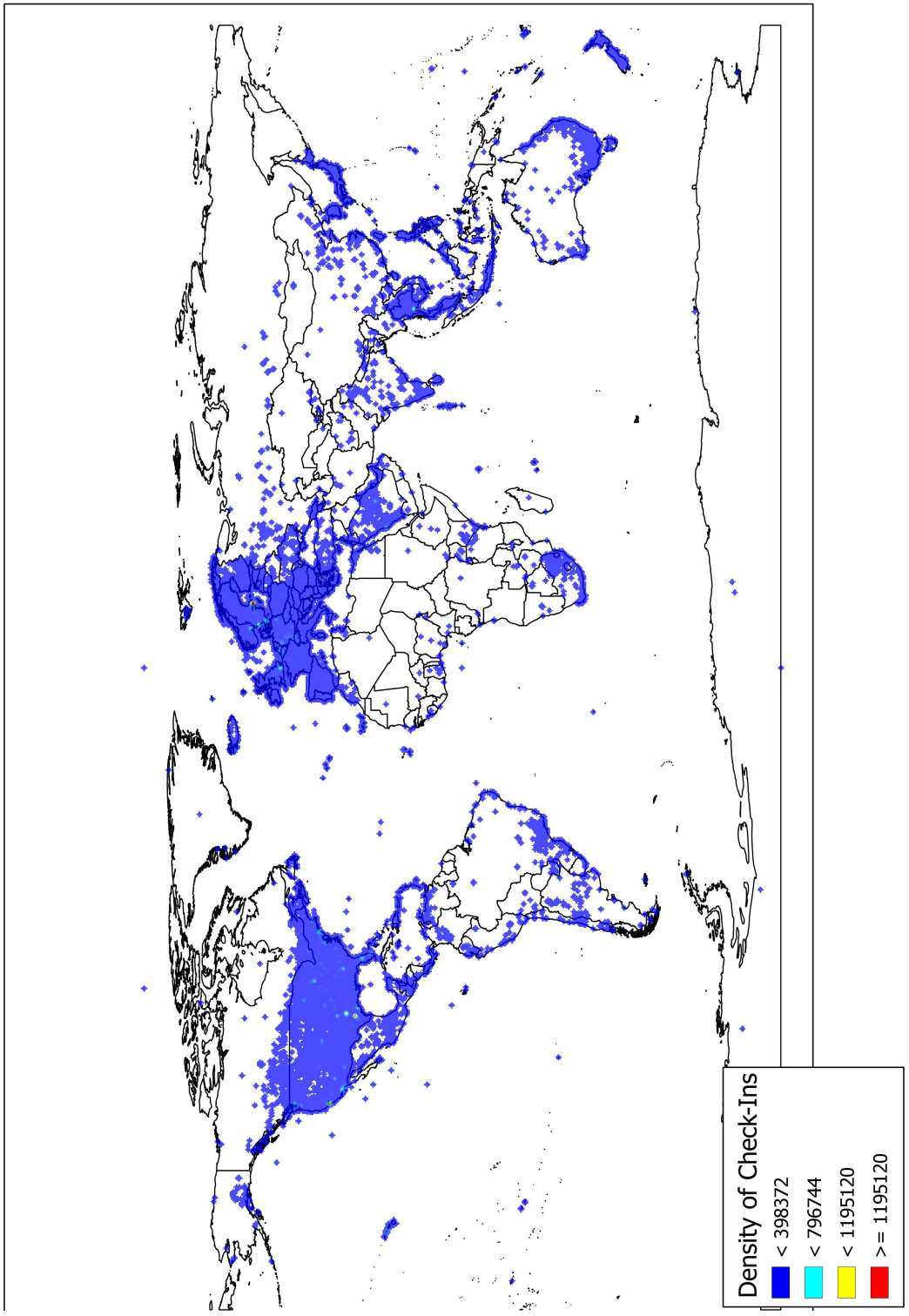


Figure 25: HeatMap of Gowalla Checkins

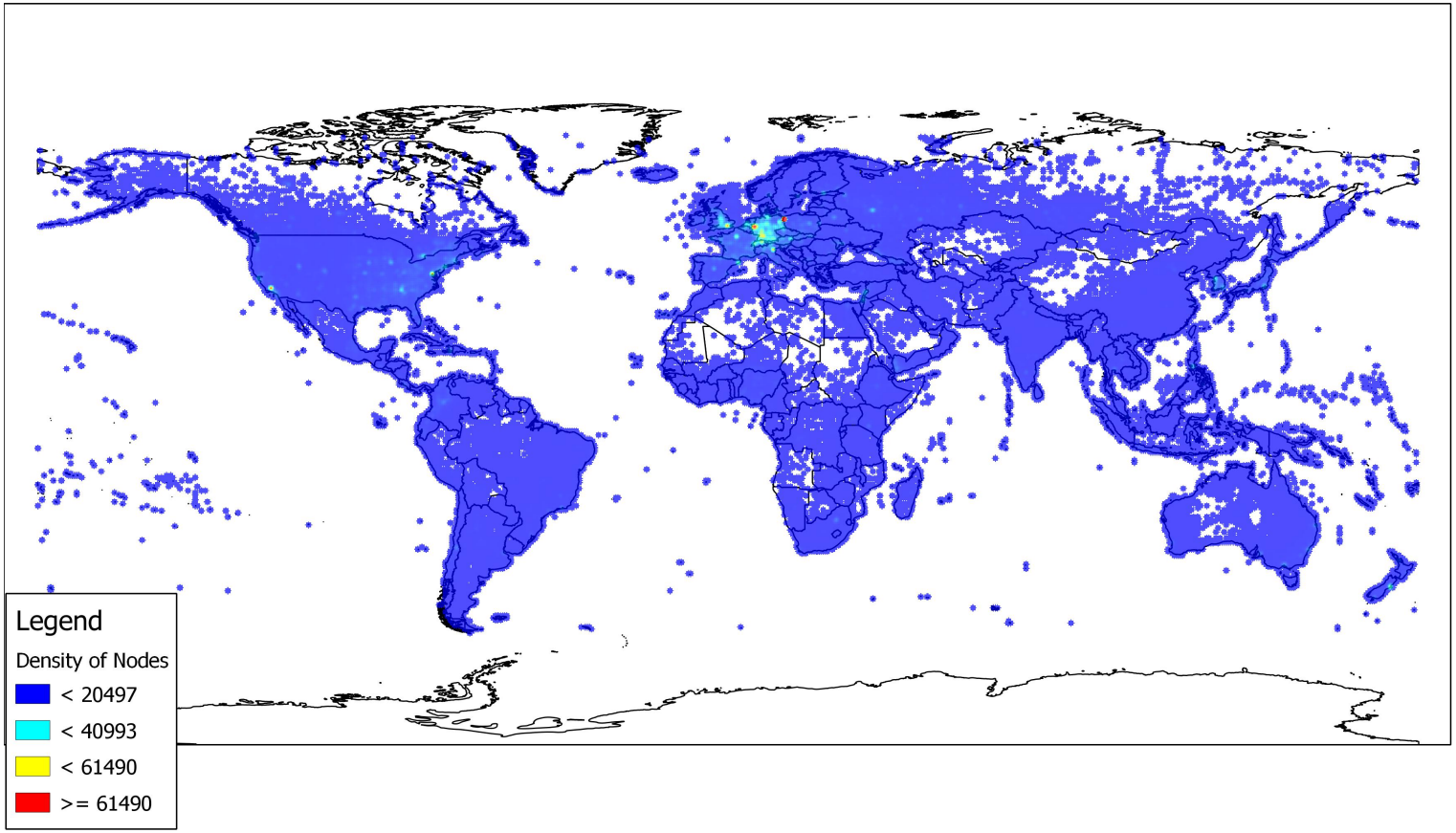


Figure 26: HeatMap of LGD Nodes

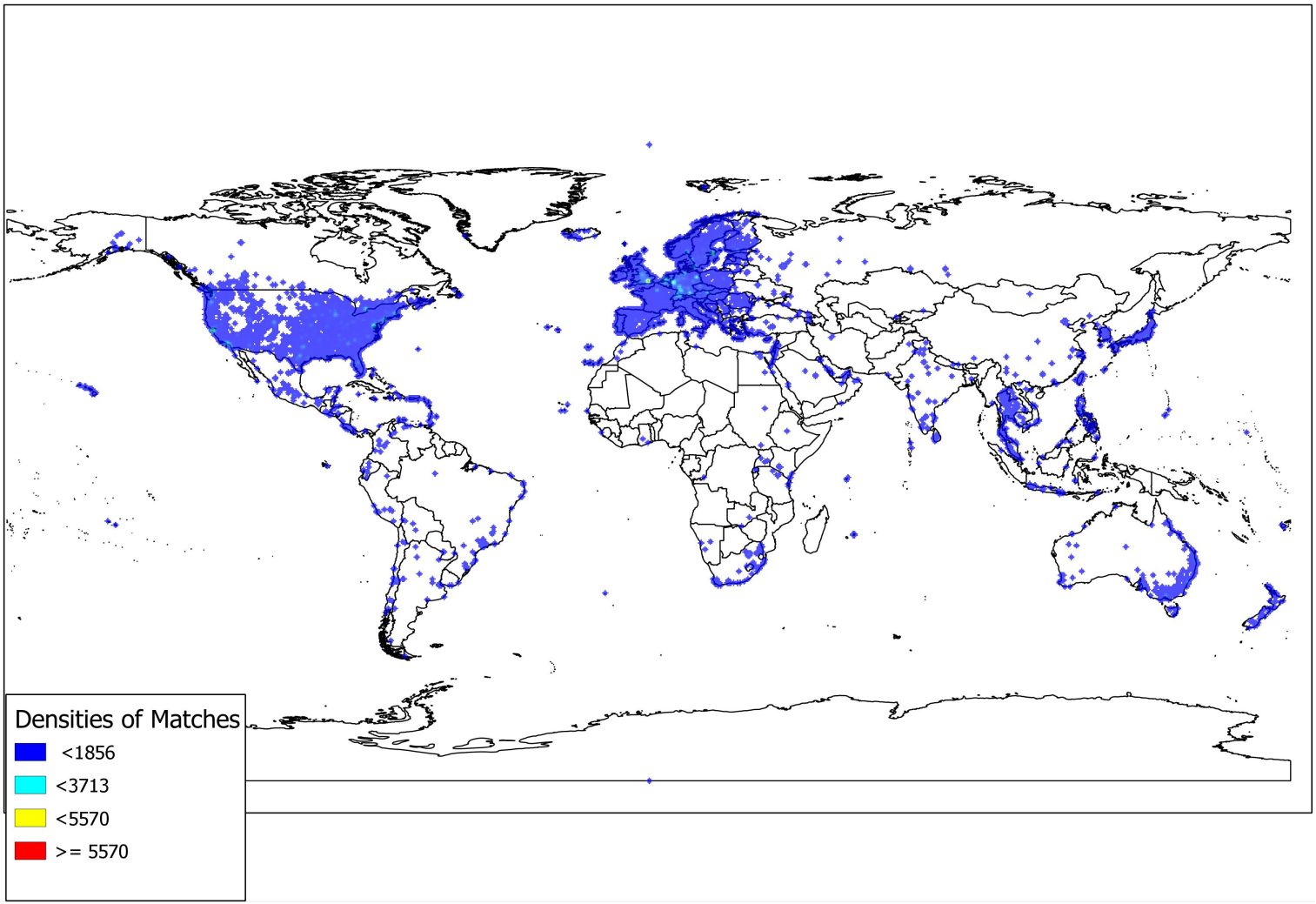


Figure 27: HeatMap of Spots with matching LGD Nodes

References

[1] Milton Abramowitz, Michael Danos, and Irene A. Stegun. *Pocketbook of mathematical functions*. Deutsch, Thun [u.a.], 1984.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[3] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145, 2005.

[4] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 217–253. Springer US, Boston and MA, 2011.

[5] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2):207–216, 1993.

[6] Alvaro Monge and Charles Elkan. The field matching problem: Algorithms and applications. In *In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 267–270, 1996.

[7] Asim Ansari, Skander Essegaier, and Rajeev Kohli. Internet recommendation systems. *Journal of Marketing Research*, 37(3):363–375, 2000.

[8] G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. Cooperative Information Systems. Mit Press, 2004.

[9] Sören Auer, Jens Lehmann, and Sebastian Hellmann. Linkedgeodata: Adding a spatial dimension to the web of data. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 731–746. Springer Berlin Heidelberg, 2009.

[10] Lars Backstrom, Eric Sun, and Cameron Marlow. Find me if you can: Improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th international conference on World wide web - WWW '10*, page 61. ACM Press, 2010.

[11] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[12] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Workshop on Context-aware Recommender Systems (CARS'09)*, 2009.

[13] Linas Baltrunas and Francesco Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems - RecSys '09*, page 245. ACM Press, 2009.

[14] V. Benjamins, J. Contreras, M. Blázquez, J. Dodero, A. Garcia, E. Navas, F. Hernandez, and C. Wert. Cultural heritage and the semantic web. In Christoph Bussler, John Davies, Dieter Fensel, and Rudi Studer, editors, *The Semantic Web: Research and Applications*, volume 3053 of *Lecture Notes in Computer Science*, pages 433–444. Springer Berlin / Heidelberg, 2004.

[15] Betim Berjani and Thorsten Strufe. A recommendation system for spots in location-based online social networks. In *Proceedings of the 4th Workshop on Social Network Systems - SNS '11*, pages 1–6. ACM Press, 2011.

[16] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: Scientific american. *Scientific American*, 2001.

[17] Betim Berjani. *Recommendation Systems for location-based Online Social Networks*. PhD thesis, Technische Universität Darmstadt, Darmstadt, 2010-12-14.

[18] George E. P. Box, J. Stuart Hunter, and William Gordon Hunter. *Statistics for experimenters: Design, innovation, and discovery*. Wiley-Interscience, Hoboken and N.J, 2 edition, 2005.

[19] danah m. boyd and Nicole B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.

[20] Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors. *The Adaptive Web*. Springer Berlin Heidelberg, Berlin and Heidelberg, 2007.

[21] William W. Cohen, Pradeep D. Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In Subbarao Kambhampati and Craig A. Knoblock, editors, *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), August 9-10, 2003, Acapulco, Mexico*, pages 73–78, 2003.

[22] Henriette Cramer, Mattias Rost, and Lars Erik Holmquist. Performing a check-in: Emerging practices, norms and 'conflicts' in location-sharing using foursquare. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11*, page 57. ACM Press, 2011.

[23] Ido Dagan, Lillian Lee, and Fernando C. N. Pereira. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1/3):43–69, 1999.

[24] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems - I-SEMANTICS '12*, page 1. ACM Press, 2012.

[25] Richard Durbin. *Biological sequence analysis: Probalistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge and New York, op. 1998.

[26] B. Furht. *Handbook of Social Network Technologies and Applications*. Springer, 2010.

[27] Sarah Jean Fusco, Katina Michael, M.G Michael, and Roba Abbas. Exploring the social implications of location based social networking: An inquiry into the perceived positive and negative impacts of using lbsn between friends. In *2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR)*, pages 230–237. IEEE, 2010.

[28] H. Gao, J. Tang, and H. Liu. Exploring social-historical ties on location-based social networks. In *Proceedings of the 6th international AAAI conference on weblogs and social media*, 2012.

[29] Gaurav Gupta and Wang-Chien Lee. Collaborative spatial object recommendation in location based services. In *2010 39th International Conference on Parallel Processing Workshops*, pages 24–33. IEEE, 2010.

[30] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[31] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 1(1):1–136, 2011.

[32] J. Horel, M. Splitt, L. Dunn, J. Pechmann, B. White, C. Ciliberti, S. Lazarus, J. Slemmer, D. Zaff, and J. Burks. Mesowest: Cooperative mesonets in the western united states. *Bulletin of the American Meteorological Society*, 83(2):211–225, 2002.

[33] T. Horozov, N. Narasimhan, and V. Vasudevan. Using location for personalized poi recommendations in mobile environments. In *International Symposium on Applications and the Internet (SAINT'06)*, pages 6 pp–129. IEEE, 2006.

[34] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[35] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, UAI'95, pages 338–345, San Francisco and CA and USA, 1995. Morgan Kaufmann Publishers Inc.

[36] Sungrim Kim and Kwon Joonhee. Effective context-aware recommendation on the semantic web. *International Journal of Computer Science and Network Security,*, 7(8):154–159, 2007.

[37] David G. Kleinbaum. *Applied regression analysis and other multivariable methods*. Brooks/Cole, Australia and Belmont and CA, 4 edition, 2008.

[38] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707, 1966.

[39] Nan Li and Guanling Chen. Analysis of a location-based social network. In *2009 International Conference on Computational Science and Engineering*, pages 263–270. IEEE, 2009.

[40] Nan Li and Guanling Chen. Sharing location in online social networks. *IEEE Network*, 24(5):20–25, 2010.

[41] Janne Lindqvist, Justin Cranshaw, Jason Wiese, Jason Hong, and John Zimmerman. I'm the mayor of my house: Examining why people use foursquare - a social-driven location sharing application. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, page 2409. ACM Press, 2011.

[42] Mohamed, Mao Ye, Peifeng Yin, and Wang-Chien Lee. Location recommendation for location-based social networks. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '10*, page 458. ACM Press, 2010.

[43] Douglas C. Montgomery. *Design and analysis of experiments*. John Wiley & Sons, Hoboken and NJ, 6 edition, 2005.

[44] Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil. An empirical study of geographic user activity patterns in foursquare. *Artificial Intelligence*, pages 570–573, 2010.

[45] K. Oku, S. Nakajima, J. Miyazaki, and S. Uemura. Context-aware svm for context-dependent information recommendation. In *7th International Conference on Mobile Data Management (MDM'06)*, page 109. IEEE, 2006.

[46] P. Garza E. Quintarelli R. Turrin P. Cremonesi. Top-n recommendations on unpopular items with contextual knowledge. *3rd Workshop on Context-Aware Recommender Systems (CARS-2011)*, 2011.

[47] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems - RecSys '09*, page 265. ACM Press, 2009.

[48] Alexandre Passant. dbrec — music recommendations using dbpedia. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *Lecture Notes in Computer Science*, pages 209–224. Springer Berlin Heidelberg, Berlin and Heidelberg, 2010.

[49] Heiko Paulheim and Johannes Fürnkranz. Unsupervised generation of data mining features from linked open data. In *International Conference on Web Intelligence and Semantics (WIMS'12)*, 2012.

[50] Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5/6):393–408, 1999.

[51] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *Lecture Notes in Computer Science*, pages 325–341. Springer Berlin Heidelberg, Berlin and Heidelberg, 2007.

[52] Prem Melville, Raymond J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *in Eighteenth National Conference on Artificial Intelligence*, pages 187–192, 2002.

[53] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer US, Boston and MA, 2011.

[54] Sarah Spiekermann and Humboldt Universität Zu Berlin. General aspects of location-based services. In J.H Schiller and Voisard A., editors, *Location-Based Services*, Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, 2004.

[55] Salvatore Scellato and Cecilia Mascolo. Measuring user activity on an online location-based social network. In *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 918–923. IEEE, 2011.

[56] Salvatore Scellato, Cecilia Mascolo, Mirco Musolesi, and Vito Latora. Distance matters: geo-social metrics for online social networks. In *Proceedings of the 3rd conference on Online social networks*, WOSN'10, pages 8–8, Berkeley and CA and USA, 2010. USENIX Association.

[57] Salvatore Scellato, Anastasios Noulas, Renaud Lambiotte, and Cecilia Mascolo. Socio-spatial properties of online location-based social networks. *Proceedings of ICWSM*, 11:329–336, 2011.

[58] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, pages 291–324. Springer Berlin Heidelberg, Berlin and Heidelberg, 2007.

[59] Mark Setten, Stanislav Pokraev, and Johan Koolwaaij. Context-aware recommendations in the mobile tourist application compass. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Paul M. E. Bra, and Wolfgang Nejdl, editors, *Lecture Notes in Computer Science*, pages 235–244. Springer Berlin Heidelberg, Berlin and Heidelberg, 2004.

[60] Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. Linkedgeodata: A core for a web of spatial open data. *Semantic Web*, 3(4):333–354, 2012.

[61] K. Virrantaus, J. Markkula, A. Garmash, V. Terziyan, J. Veijalainen, A. Katanosov, and H. Tirri. Developing gis-supported location-based services. In *Proceedings of the Second International Conference on : Web Information Systems Engineering, 2001*, volume 2, pages 66–75 vol.2, 2001.

[62] Rui-Qin Wang and Fan-Sheng Kong. Semantic-enhanced personalized recommender system. In *2007 International Conference on Machine Learning and Cybernetics*, pages 4069–4074. IEEE, 2007.

[63] William W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[64] Mao Ye, Krzysztof Janowicz, Christoph Mülligann, and Wang-Chien Lee. What you are is when you are: the temporal dimension of feature types in location-based social networks. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '11*, page 102. ACM Press, 2011.

[65] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik-Lun Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, page 325. ACM Press, 2011.

[66] Liyang Yu. *A developer's guide to the semantic web*. Springer, Heidelberg and and New York, 2011.