Enhancing a Location-based Recommendation System by Enrichment with Structured Data from the Web

Max Schmachtenberg
Universität Mannheim
Schloss
Mannheim,Germany
max@informatik.uni-mannheim.de

Thorsten Strufe
TU Darmstadt
Hochschulstraße 10
Darmstadt, Germany
strufe@cs.tu-darmstadt.de

Heiko Paulheim
Universität Mannheim
Schloss
Mannheim,Germany
heiko@informatik.uni-mannheim.de

ABSTRACT

Location-based social networks (LBS) enable users to check-in at points of interests (POIs), share this information with other users within the network, and receive recommendations about new and interesting POIs in their vicinity. In this paper, we show how such recommendations can be improved by adding background information from Linked Open Data and other sources of structured data. Using a dataset previously crawled from the LBS Gowalla, we analyze which types of background information are the most beneficial for improving the recommendations. In a series of offline experiments, we show that the quality of recommendations can be improved by 51% in precision and 150% in recall.

Categories and Subject Descriptors

Information Systems [**Retrieval tasks and goals**]: Recommender Systems; Networks [**Network Types**]: Social media networks

General Terms

Algorithms, Experimentation

Keywords

Linked Open Data, Recommender Systems, Location-Based Social Networks

1. INTRODUCTION

Location-based Social Networks (LSNs) combine capabilities of online social networks, for example to create profile pages, establish friend connections to other users and share information about oneself, with those of location-based services, i.e. services that integrate a device's position with other information. The result is a service that allows users to perform check-ins, thereby stating that they are at a given POI. This information can then be published on their profile page and shared with friends, other users, or the public, depending on the LSN and the user's privacy settings.

WIMS'14, June 02 - 04, 2014, Thessaloniki, Greece

http://dx.doi.org/10.1145/2611040.2611080.

An important service offered by LSNs is to give users recommendations about POIs in their vicinity, which can foster the service's usage, as it allows users to discover locations they are interested in and were not aware of before. Such recommendation systems can be content-based, in which case they leverage properties of POIs for which the user previously expressed his preference. If, for example, a user often shows his liking for POIs of category restaurant, it might be useful to recommend POIs with the same category to him. To improve the model of a user's preferences, additional knowledge about the POIs might be helpful. If, for example, the menu of restaurants was available, it might turn out that the user's rating also depends on the dishes offered. By having more information about the POIs to be recommended, a more nuanced impression of a users preferences can be created, enabling better recommendations.

The same is true for the context of a check-in. Knowledge about the circumstances of a check-in, for example the time of day or the weather, may yield that a user does not have a utility for restaurants in the morning, or that he does not like parks while it rains. With this knowledge, a recommendation system hence can give recommendations adapted to the context under which recommendations are requested. Possible sources for additional knowledge in the form of structured data are readily available on the web, e.g., Linked Open Data (LOD).

In this paper, we evaluate how to improve recommendation for a LSN called *Gowalla* by adding structured knowledge from *LinkedGeoData* (LGD), a Linked Data version of *OpenStreetMap* (OSM), and *MesoWest*, which offers weather information about North America in structured form. Using these sources, we add knowledge to POIs and check-ins from Gowalla and generate additional features from Gowalla data itself. With these sets of additional knowledge, we evaluate the effects of different kinds of additional knowledge on POI recommendations with a series of offline experiments, using different classifiers, user groups, combinations of additional knowledge and recommender types.

The rest of this paper is structured as followed. The next section gives an overview over LSNs and their core properties as well as on recommender systems, formalizing their concept and describing different approaches. Section 3 gives a brief overview of the related work, both with respect to LSNs as well as recommender systems. In Section 4, we de-

scribe Gowalla, the LSN on which our experiments are performed, and the dataset crawled from it as well as LGD and MesoWest. Section 5 describes how we augment knowledge about Gowalla's POIs with knowledge from LGD, and about check-ins with knowledge from MesoWest. It also covers descriptions of the resulting groups of additional knowledge created for POIs and check-ins. In Section 6, we present our experimental setup, describing how we perform offline recommender experiments, while varying the knowledge added, learning algorithms and user groups. The results are presented in Section 7, elaborating under what circumstances additional knowledge is beneficial for recommendation. We close the paper with summarizing our results as well as pointing to possible enhancements for the future.

2. BACKGROUND

The last years saw multiple developments that lead to the emergence of LSNs. First, Online Social Networks (OSNs) saw a strong growth. The biggest one, Facebook, reporting 1.2 billion monthly active users as of January 2014. Second, mobile devices, utilizing mobile broadband and having the ability to use location services such as GPS or WLAN triangularization, saw a wide distribution. By now, more than 500 million users access Facebook from such mobile devices, and the number of mobile Facebook users outnumbers those accessing the OSN with desktop PCs.[1] The combination of such mobile broadband subscriptions and the ability to determine a device's position using location services led to the emergence of so called location-based services, where the information received from it are adapted to the location of the device.

Combining these developments, location-based social networks (LSNs) such as Foursquare, Gowalla, Google Latitude, and Facebook Places[2] emerged.[3] The services offered by such LSNs typically include:

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

When performing a check-in, a user has to be in the vicinity of the POI, which is determined using location services. If a POI where a user wishes to check-in does not exist, he can usually create it himself. He can also share the information of his check-in with friends, other users of the LSN and the general public, depending on the LSN and its privacy settings. To keep users involved, LSNs often include gaming features, where users can earn badges for performing certain actions. For example, a badge can be given to perform a number of check-ins at a given location, aiming to foster competition with other users. Lastly, users can receive recommendations about POIs in their vicinity based on their preferences. By helping the user to explore his vicinity and find places he likes and did not visit before, the value of the LSN to the user and thus his usage of the service is strengthened.

Such recommender systems are based on the idea that there exists a utility function, were every user assigns his utility, expressed as a rating, to an item, such as a POI. When considering the context of a situation a user assigns such a rating, for example time, a place, or something else, this rating becomes depending on it, for example a user having different utilities for a venue at different times of day.

The ratings a user assigns to POIs may be scalar, for example a rating from zero to five, binary, expressing that the user likes or does not like an POI, or unary, where the user expresses that he likes a POI, while no expression of preference means no information is available. Additionally, these ratings can be explicit, were the user actively expresses his utility of a POI, or implicit, which means that his opinion on the POI is derived from his behavior, for example if he visited a POI or not. While the first is more precise, as the utility does not need to be derived from a user's behavior, data can be collected more easily and quickly with an implicit approach.

As typically for only a small fraction of items a rating from a user exists, the task of a recommender system is to extrapolate the rating for those were no rating is available. [4] distinguishes between three kinds of basic recommender systems (i.e. they involve no context): collaborative, content-based and hybrid. When using collaborative methods, the rating of an item is derived from the ratings assigned by similar users. Content-based methods derive the preference from ratings the same user gave to similar items. Hybrid methods combine these approaches, for example by combining example predictions from collaborative and content-based methdos, incorporating collaborative characteristics into a content-based approach, for example by creating features for an item describing how well other users liked it.

An enhancement of these basic models are context-based recommender systems. They include the circumstances of a rating and thus can also give recommendations depending on the context of the situation. A user might for example have a lower utility for a restaurant in the morning than in the evening, as he does not usually visit restaurants at the beginning of the day, thus leading to a lower (implicit) rating for a restaurant given the context of time. Here, the concepts pre-filtering, post-filtering and contextual modeling exist. For contextual pre-filtering, a subset of users and/or items are selected, and traditional recommendation is performed. Post-filtering on the other side takes the recommendation of a traditional recommender and adapts it to the context, for example by adapting the ratings or filtering out items. Lastly, contextual modeling creates models which natively include context, for example by working on an multidimensional matrix of user/item/context ratings for collaborative filtering.

3. RELATED WORK

Recommender systems have been researched intensively over the last two decades [2, 17]. For the more specific problem of POI recommendation, previous approaches prior to the emergence of LSNs tried to record user behavior of test users in field experiments, leading resulting in a scarcity of training data with comparatively few users and check-ins. (see for example [12, 18]). With the emergence of LSNs, more abudant data became available, with millions of users

[1] `http://investor.fb.com/results.cfm`

[2] By now, Gowalla and Latitude have being shut down

[3] Although Facebook for example is also an OSN, one can see it as a LSN, as it offers the functionality of LSNs, even if it is not the most central part of it.

and POIs. Such approaches for POI recommendation usually utilize collaborative filtering, where the spatial aspects and, in case of LSNs, the friend aspects are exploited to improve recommendations or to decrease computational cost. For collaborative filtering, an approach to ease computational costs by focusing on users that have the same spatial activity pattern that the user for which recommendations is given ([13, 10]). Pre-filtering may also be applied to the spots in order to only evaluate the rating of those that are in the vicinity of a user [12]. Furthermore, model-based collaborative filter approaches derive a rating and apply regularized matrix factorization [7].
The inclusion of contextual information into the recommendation process has also been explored, albeit not in the context of POI recommendation. A general apprach is to apply pre-filtering, where the n-dimensional space of items, users and context is segmented, and for each segment, seperate recommenders are used ([1]). [5] for example segments depending on time, creating sub-profiles of users for different time contexts. On the other side, [6] perform such a splitting of profiles with items, also depending on context of time. A general downside of such methods is that the sparsity of rating matrices is increased, as more items are available for the same number of users and ratings. [15] give example of post-filtering approaches, were they change the ratings of recommended tiems based on the probability of a context or filer them out of the recommended item list if they are too inprobable given a context. Lastly, [14] try to extend a SVM to incorporate contextual dimensions for restaurant recommendation. They include additional variables such as time, a user's budget, properties of the resturant as well as properties such as time or weather. The extended collaborative filtering approach was then applied in a small scale experiment on a low number of users and restaurants, showin that such approaches can enhance the recommender's accuracy and the user's satisfaction with the recommendations.

4. DATASETS

The basis of our experiments is a dataset from Gowalla, a location-based social network founded in 2007. Gowalla was bought by Facebook in 2012 and shut down consecutively. The dataset was crawled from the Gowalla API from June to October 2010 [8]. It contains data about 1.72 million POIs, called *spots*, with additional information such as a unique spot ID, name, the user ID of the spot's creator, creation date of the entry, city and state it is located in, aggregated statistics, such as number of check-ins, unique visitors and photos associated with the spot, geographic coordinates and the range at which a user is allowed to check-in, a textual description, Twitter ID and website associated with it, as well as a category.

Besides data about spots, the dataset contains information about 177,000 users with information such as a unique user ID, first and last name, age, hometown, a short biography, Twitter and Facebook ID, aggregated statistics such as the number of stamps, photos, pins, items and confirmed friends and links to other users they are friends with. In total, the dataset records 16.56 million check-ins, where each check-in relates a user and a spot, and is marked with a time stamp.

The majority of all spots (48.41%) are located in the United States, where also the majority (54.61%) of all check-ins is performed, followed by Sweden (9.11% of all spots, 11.77% of all check-ins), Germany (6.6% spots, 5.65% check-ins), and the United Kingdom (5.09% of all spots, 4.11% of all check-ins). Spots and check-ins are usually massed at urban areas, the most prominent ones being San Francisco, Los Angeles, New York, Austin and Dallas in the U.S., Stockholm and Gothenburg in Sweden, Berlin, Munich, the Rhine-Main Area and Cologne in Germany and London and Manchester in the United Kingdom. Of all spots, 73% have a category assigned, the biggest category being "Gas & Automotive" (4%), "Asian Food" (3%), "Corporate Office" (3%), "Other" (2%) and "Grocery" (2%)

From the Linked Open Data Cloud[4], we identified LinkedGeoData (LGD)[5] as a suitable candidate to add knowledge about spots from Gowalla. Developed by the University of Leipzig, it is based on Open Street Maps (OSM), a collaborative mapping project [3]. The data was converted to Linked Data using an ontology derived from tags with which entities in OSM are annotated. This results in a dataset containing 6.2 million POIs, called *nodes*, described with 65 million triples. The classes with the largest number of nodes, specified by [19] are "parking"(8.3%), "village"(8.2%), "shop"(7.9%), "hamlet"(6.6%), "school"(5.7%), "PlaceOfWorship"(5.7%), "restaurant"(2.7%), "pub"(1%), and "fastfood"(1%). Comparing this ranking to the list from Gowalla, one can see that the categories of POIs differs in the two data sets, with categories "parking", "village", "hamlet", "PlaceOfWorship" being places that one would usually not consider for a check-in by Gowalla users.

Information associated with a LGD node can be very heterogeneous, as there are no strict requirements which information about a node has to be entered. A node at least contains a name, its type, and its geographic coordinates. Additionally, it may also contain the country and city the node is in, an address or additional information such as wheelchair accessibility or offered goods. The highest density of nodes can be found in Europe, especially Germany, which has a general high coverage of OSM data.

To add contextual knowledge to Gowalla's check-ins, we use data from MesoWest[6]. This project from the Department of Atmospheric Sciences at University of Utah bundles weather information from various U.S. weather observation networks, providing unified access to the information and additional metadata, with harmonized quality control and data formats [11]. More than 200,000 stations from 176 different networks can report up to 150 variables, depending on the station's equipment. Basic variables always reported are temperature, humidity, air pressure, or dew point. Additionally, information about the station itself, such as its geographic location, the station's surroundings, and the sensors installed is available. Both weather as well as station information can be retrieved in various formats, including XML, spreadsheet or CSV. Including mostly U.S. weather station networks, weather information from MesoWest is only available for the U.S. and some regions of Canada.

5. AUGMENTATION OF POIS WITH BACKGROUND KNOWLEDGE

Background knowledge about Gowalla entities can be added

[4] `http://www.datahub.io/group/lodcloud`
[5] `http://linkedgeodata.org`
[6] `http://mesowest.org`

using the datasets described above. Nodes from LGD which describe the same POIs like spots from Gowalla can serve as a source for additional information, as they may contain information about the POI not available in Gowalla. Also, nodes from LGD may be used to describe the environment of a Gowalla spot, helping to capture the nature of the neighborhood of a spot. This has the advantage that knowledge from LGD can be added to a spot from Gowalla, even if no corresponding node could be found. Using the timestamp of a check-in and the position of the POI where it occured, data about the weather condition prevalent during a check-in can be added from MesoWest, adding knowledge about the context of a check-in.

5.1 Linking Gowalla and LinkedGeoData

To link POIs and nodes from Gowalla and LGD directly, we try to find for every spot from Gowalla a corresponding node from LGD. For this task, we restrict ourselves to one to one relationships of POIs. A shopping mall, containing multiple stores, might be represented by many different stores in Gowalla, as their owners want users to check-in at their store, while in LGD, only a node for the mall itself may exist, as it is only interesting as one big complex from a cartographic point of view. By ignoring such one to many relations, we avoid adding a large degree of complexity.

For matching spots and nodes, we identified three properties existing in both datasets which we deem comparable: name, location and category.

Both datasets offer a name for the POI described, which one would expect to be identical. But as the two datasets have been created by crowdsourcing processes from different groups of users, name variations are inevitable. Variations often encountered are for example to omit articles, e.g. "The Cheesecake Factory" and "Cheesecake Factory" or to have varying words around a proper noun, for example "The Wheatsheaf and "Wheatsheaf Hotel". Typing errors on the other side were encountered only rarely, which might be due to the fact that both datasets undergo some sort of quality control.

The position of POIs, expressed as longitude and latitude, is also available in both datasets. Again, differences can be observed, possibly due to imprecision of the devices used to record the positions, or variations in the measurements of the position, for example by stating the position of the entrance of a shopping center versus its central plaza.

Lastly, both datasets offer categorizations of POIs. While Gowalla allows up to one category per spot, taken from a catalog of categories, LGD allows for multiple tags, taken from a community-maintained set of tags.

To automatically link entities of the two datasets, we created a heuristic which takes the similarity of the mentioned properties into account. For measuring the performance, as well as to have a training set, we created a gold standard by manually linking entities. We focused on spatial areas deemed promising to find many nodes that would match with spots, namely larger urban areas in Germany, the US, the UK, France and Sweden, which also includes spot names in multiple languages. Sampling was performed by taking all spots 30 kilometer near the town's center[7], sampling a random number of initial 50 spots from Gowalla and searching for corresponding nodes by retrieving the list of nodes that are in the spot's vicinity, picking the one we considered to represent the same POI. Analyzing 435 spots, we linked 104 with corresponding nodes.

Based on the matches and spots without matches in the gold standard, we developed a mapping heuristic, utilizing name, location and category of spots and nodes.

5.1.1 Matching by Name

To evaluate the similarity of the POIs' names, denoted in the following as two strings a and b, various name comparison metrics exists, which can be categorized into edit-like distance functions, token-based functions, and hybrid functions [9]. Edit-like distance functions measure how strongly string a has to be edited to be identical to string b. One of the most well-known metrics from this category is the Levenshtein-distance function. Here, the basic operations are the insertion, deletion, and substitution of a character. The Levenshtein distance is then computed by the number of operations it takes to transform string a to string b. It can also be normalized, using the maximum number of changes needed to convert a string with the length of string a to string b. Another example of such metrics are the Needleman-Wunsch distance and its variants, using a dynamic programming approach to find alignments between two sequences. Lastly, the Jaro and the Jaro-Winkler distances are based on the number of characters of a string a that are at similar positions compared to string b and the order of these characters.

A problem of edit-like distance functions is their sensibility to the omission and order of words within a string, like in the examples described above. Especially the latter variation makes it necessary that the string is heavily edited, lowering the similarity of edit-like distance functions significantly. As other variation, such as typos, are comparatively infrequent, the Needleman-Wunsch metric, which performed best of all edit-like functions, had a maximum F1 score of 0.793 with an average precision of 0.453 on our gold standard.

Token-based functions are based on splitting up a string into chunks of characters (tokens) of various size and comparing the resulting token sets with each other. Well-known metrics are for example the Jaccard distance, TF-IDF, or the Jensen-Shannon distance. Of these metrics, the latter two require a corpus of tokens from all strings for reference. For this, we used the names from spots of our gold standard as well as all nodes that are located in an area 300 meters around the spot.

The Jaccard distance compares the quota of common tokens of string a and b with the union of their token sets. It yields better results than edit-like distance metrics, with an maximum F1 score of 0.836 and average precision of 0.743. Problems with this metric arise for strings like "Blåbär Café" and "Blåbär" where the more expressive (because less frequent) word "Blåbär" is present in both strings, while the widely used word "café" is missing, resulting in low similarity of 0.62 by the Jaccard metric.

To balance this shortcoming and take the expressiveness of words into account, TF-IDF (Term Frequency - Inverse Document Frequency) might be used, where the importance of a token is inversely proportional to its appearance in the set of all strings. While TF-IDF improves results for the example above, it may overestimate the similarity two names where the difference lies in tokens that appear frequently in all names, but which are an important addition to the

[7]Taken from Wikipedia

more unique word of a string. For example "Ringcenter" and "Kiosk am Ringcenter" have a high similarity in TF-IDF, as "Kiosk am" yields tokens that are relative frequent, while "Ringcenter" is relatively unique. In our experiments, TF-IDF yields a maximum F1 score of 0.858 with an average precision of 0.72.
Finally, the Jensen-Shannon (JS) distance interprets tokens from two names as probability distributions, which can be compared using the Kullback-Lieber divergence. To consider for variations between two strings, different smoothing methods may be applied, like JelinekMercerJS, or DirichletJS, which performs Bayesian smoothing using a Dirichlet Prior. In the implementation used, the unsmoothed version was based on words of a string. This and the non-existing smoothing resulted in relative low performance of a maximum score F1 of 0.76 and average precision of 0.64. JelinekMercerJS and DirichletJs on the other side showed high maximum F1 scores, but generally lower average precisions of both 0.65.

Lastly, hybrid distance metrics combine edit-like distance functions with token-based functions. The Level2 (L2) metric uses a secondary function, like an edit-like distance function, to compare tokens of two strings by averaging the maximum possible similarity of a token from string a when compared to the tokens of string b. Alternatively, soft TF-IDF may be used. Unlike the "hard" version of TF-IDF described above, it compares not only matching tokens, but also "similar" tokens up to a predefined threshold, evaluated by an inner function. The similarity of two tokens are then additionally considered when computing the similarity.

When testing hybrid methods using edit-like distance functions as inner functions, they yielded results slightly better than token-based functions in terms of maximum F1 score, with a comparable average precision. The best hybrid function was Level2 using the scaled version of Levenshtein, with a maximum F1 score of 0.878 and an average precision of 0.743.

5.1.2 Matching based on Geographic Distance

Another way to assess the similarity between a spot and a node is to look at the distance between the two, computed from the geographic coordinates of the two POIs. As we observed that all pairs of spots and nodes had a distance less than 300 meters, we defined a spatial similarity value of one to a distance of zero meters and a metric value of zero to a distance of 300 meters and above.

One can also expect that the distance between many mapped spots and nodes is close, but not equal to zero. This is a result of measurement inaccuracies, which means that the function mapping the distance to a similarity score between 0 and 1 should not be linear, but convex and monotonically decreasing. To find such a curve, we used a quadratic Bézier-Curve. Such a curve is defined by a set of control points, where the first and last one are the starting point and endpoint of the curve, i.e. $(0, 1)$ and $(0.3, 0)$ in our example. The third, intermediate control point is then used to define the shape of a polynomial, defining a polynomial of degree two. While varying the position of this point, one can control the shape of the curve.

By systematically trying all positions of the control point in the rectangle defined by the other two control points, we found the optimal shape of the curve. With the control point at position $(0.06, 0.4)$, it yielded a maximum F1 score of 0.365 with an average precision of 0.17.

5.1.3 Matching based on Categories

Lastly, one can compare categories of spots and types of nodes. Since both datasets use a different set of categories, they are not directly comparable. To match the category systems, we create association rules between categories of spots and types of nodes. To learn such rules, our manually created gold standard was too small to mine meaningful association rules. By using the knowledge previously gained, we thus automatically create a reliable training set. This was done by automatically matching all spots and nodes that have 1) exactly the same name and 2) a distance less than 50 meters. Using this technique, we were able to find 59,617 matches, or 3% of all available spots, which we can use to create association rules, using the Apriori implementation by Christian Borgelt.[8] Allowing only Gowalla categories as antecedents and only LGD categories as consequences, using the default settings, and posing no lower bound on support, a total of 4,145 rules were created. Using these rules, the metric alone yielded a maximum F1 score of 0.365 with an average precision of 0.17.

5.1.4 Hybrid Matching

Finally, the metrics outlined above were combined into a unified metric, using a linear framework. For every metric, a weight is assigned, with the sum of weights normalized to 1. As all metrics are also normalized, we receive a similarity score between 0 and 1 for every potential mapping. For string similarity, we tried different token-based and hybrid similarity measures within the unified framework to find the optimal metric and associated weight. The best results were achieved using TF-IDF as string metric, and by assigning a weight of 0.5 to the name metric, a weight of 0.4 to the distance metric, and a weight of 0.1 to the category metric. This resulted in a maximum F1 score of 0.929 with an average precision of 0.7.

Using this metric, we were able to find 176,468 matches, or 10.24% of all spots, accumulating 2.6 million check-ins, which is 16% of all check-ins. The highest density of matches can be found in the London area, followed by matches in Germany. While a considerable number of matches were found in bigger U.S. cities, the density is low compared to all spots in these cities.

5.2 Feature Generation from LinkedGeoData

The matches between Gowalla and LinkedGeoData can be used to generate features for the matched spots. We used the tool *FeGeLOD*, which allows to perform unsupervised generation of data mining features from Linked Open Data [16]. FeGeLOD offers six strategies for generating features. Two of them are entity based, one creating features from data properties of a resource, while the other one creates features from the type definitions of an entity. The other four strategies take relations between the entity and others into account. Since entities in LinkedGeoData are rarely interlinked, we only extracted type definitions and data properties.

With this, we could add the categoy of a linked node to a spot, The data properties added included information like cuisine of restaurants, additional address information,

[8] `www.borgelt.net/apriori.html`

wheelchair accessibility or links to resources representig geographical entities containig the spot.

5.3 Description of a Spot's Environment

Since a direct mapping of Gowalla spots and LGD nodes is only possible for roughly a tenth of all Gowalla spots, we need additional means to augment the other spots in order to make meaningful recommendations. To do so, we also utilize data from LGD for enhancing knowledge about Gowalla spots by using the type description of nodes which are located near spots to describe their surrounding. Based on a spot's environment, user preferences may be captured, for example that a user avoids restaurants in (possibly crowded) shopping areas. Compared to directly utilizing knowledge from corresponding nodes, this technique has the advantage that it can be applied to every spot, not just the ones where a match was found.

To exploit such information, we retrieved the types of all nodes within a radius of 100 meters around the spot's locations. This leads to an average of around five different types of nodes near a spot, which are collected as a vector describing the spot's surroundings.

5.4 Feature Groups

The approaches described above led to two different kind of additional feature groups (FGs) for spots. To contrast these feature groups with knowledge derivable from Gowalla itself, we created two additional feature groups, i.e., friend features and the environment of a spot described through other Gowalla spots in its vicinity. Thus, we utilized the following feature groups in our experiment:

FG0: spotfields: This feature group includes all 16 attributes available for spots in Gowalla, as outlined in section two. The textual description of spots was converted into a bag of word representation, using a unary tokenizer with on average 5.3 features.

FG1: friend features: We created two features describing the importance of a spot to the friends of a user. One feature describes the fraction of the spot's check-ins done by a user's friends while the other states if the spot was created by a friend of the user. As these features are computed for every user individually, depending on his friends, one can see them as hybrid recommender features, as they capture the opinion of a user's friends about a place.

FG2: Gowalla environment: Similar to the environment of LGD nodes, this feature group captures the number of spots of a certain category that are in the vicinity of a given spot. For every spot, we created features indicating if a spot of a given category is in the spot's vicinity, enabling us to compare the results for the LGD environmental features to the ones from Gowalla to see which environment is more helpful for recommendation. It contains a total of 388 features.

FG3: LGD environment: This feature group consists of all types of the nodes that are in the vicinity of a given spot. Similar to before, a feature indicates that a node of a certain category is in a spot's vicinity. This group contains 5,314 features.

FG4: node data: This feature group consists of features generated by FeGeLOD for spots where a matching node was found, creating 342 features. The textual features were converted to a bag-of-word representation with an average of 8.67 features. If a spot could not be matched to a corresponding node, all features have missing values.

5.5 MesoWest Weather Data

To annotate the check-ins with data from MesoWest, we first linked spots to nearby stations and then crawled data for the times of check-ins. Linking spots to station was done by searching the nearest station within a 30 kilometers radius of the spot. We restricted ourselves to networks with more than 500 stations and to stations which were setup before the first check-in occurred, as to lessen the required number of necessary API calls. This resulted in finding nearby stations for 45% of all spots, a number similar to the quota of all spots located in the U.S., showing that for the majority of spots, stations are nearby.

As the API returns information for a 24 hour period, we then created a list of all days for every station when a check-in occurred at a neighboring spot. These information were then crawled and finally the weather information were linked to the check-ins.

The resulting weather information contained 32 variables, describing 18 physical variables. As the equipment of stations differ, variables were not reported by all stations. We excluded variables not reported for at least 60% of all check-ins. This results in 12 variables, including those reported for all check-ins, temperature, wet bulb, dew point, heat index, and wind chill.

To evaluate context, we compared the context of weather with time information. In this approach, we segmented the timestamp of check-ins, creating features like the time of day, the day of the week or the month.

6. EXPERIMENTAL SETUP

With different combinations of feature groups and the possible addition of contextual information, we performed experiments to evaluate with which additional knowledge, recommendation performance improves.

In our experiment, we emulate real-world use cases. If a LSN recommender system gives recommendation to a user, it usually also gives a list of recommendations of spots in his vicinity. A user is in an area with different spots, chooses one, and performs a check-in there. Thus he votes in favor of ("visited") one spot and decides against the other ones ("not visited") in its vicinity. Such a statement of preference constitutes an implicit binary rating.

6.1 Experiment Data

With this basic notion of a check-in situation, we perform offline recommender experiments. Such experiments are performed by taking a large fraction of known transactions, check-ins in our case, as training data to learn a user model, while the remaining data is used for evaluating the model. Although no real users are involved and thus the interaction between recommender system and users is excluded, they allow to compare recommendation approaches at low costs.[9] As timestamps are available, it suggests itself to train on older data and test on newer data, which is what would also happen in reality. For every user, we thus used the first 80% of all check-ins of a user as training data, and the last 20% as test data.

[9]See [17] for a discussion on recommender system experiments

The fact that a user chooses one spot and decides against the others in his vicinity has to be represented in the data. For simplicity, we assume the area near the user to be one kilometer around the spot the user checked-in, thus including all spots in this area as spots the user considered. This is certainly only an approximation, as an area of one kilometer does not necessarily include spots the user might have in mind (the area might also be bigger or smaller). Thus for training, we presented to a learning algorithm the spots the user visited as "visited" (one) and spots he did not visit as "not visited" (zero). Likewise, when evaluating the recommender, we presented a list of spots around the spot the user visits to the recommender system, which labels the spots as zero or one.

For evaluating the influence of background information from LGD in more detail, we performed experiments with two user groups, in the following referred to as *experiment one* and *experiment two*. To have enough numbers of check-ins, we only sampled active users, i.e. users that have at least fifty check-ins. Experiment one uses a group of 1,000 randomly sampled active Gowalla users, while experiment two uses a randomly sampled group of 1,000 active Gowalla users where the quota of check-ins at nodes that can be mapped to LGD is higher than 0.5.

6.2 Algorithms and Feature Groups

For evaluating the performance of recommendation when using knowledge added from LGD to Gowalla, we performed experiments with content-based recommendation. From machine learning algorithms frequently used [17], we chose to use Naive Bayes and Ripper, as representatives of different types of algorithms, both being known for offering a good trade-off between accuracy and runtime.

For adding knowledge to spots, we had a total of four additional feature groups, in addition to the baseline group zero (see above). For every of the sixteen possible combination of additional feature groups, we created models using both algorithms for both experimental groups, leading to a total of 64 models. These were then evaluated with either all check-ins from the test set or only for check-ins at spots the user did not visit before, i.e. check-ins of spots where no check-in was performed which can be found in the training set.

To use context information, i.e., weather data from MesoWest, we applied post-filtering, which enabled us to re-use the results from the previous content-based recommendation and adjust them to the given context. We followed the approach described by [15]: For a given list of recommendations, we re-weighted the results by either recommending a previously not recommended spot if the probability of the context is above a given threshold or by not recommending a previous recommended spot if it is below a given threshold, making it necessary to find suitable threshold for every model.

As we described above, we apply the contexts of time and weather, which we evaluated separately.

6.3 Metrics

As metrics for evaluation, we used precision and recall. Due to the fact that only one spot is visited for every check-in, precision is either zero if this spot is not recommended, or one divided through the number of not visited, but recommended spot if the visited spot has been recommended. Thus the main goal is to maximize precision, which has the main focus of our analysis.

7. RESULTS

As a baseline, we first evaluated the performance of Naive Bayes and Ripper without any background knowledge, as seen in Table 1. It can be observed that a) Naive Bayes outperforms Ripper, b) the recommendations for experiment two are better (i.e., background knowledge helps in the predictions), and c) the predictions for existing spots are worse than those for new spots. The low precision values are partly caused by our experimental setup: as we use a candidate set of all spots in the area of one kilometer around the user's locations, out of which only one is considered correct, false positives are almost inevitable.

7.1 Influence of LGD Features

For the addition of LGD features, we will first analyze the results occurring with Naive Bayes as a learning algorithm. Taking a look at Figure 1, we see that precision and recall are generally improving when adding feature groups, the only exception being model 01 and 012 for experiment two, lowering average precision, and models 02, 03, 023 which have lowered average recall.

Next, we take a look at the effects of individual feature groups by comparing the average improvements of models using a feature group with the average improvements of those not using the feature group.

Feature group one shows with 1.45% and -1.51% an increase for experiment one and a decrease for experiment two in precision, while average recalls are more similar with 3% for experiment one and 2.25% for experiment two. This can be seen as the result of the number of candidate spots for check-ins in experiment two being generally larger than for experiment one. Thus, more spots are recommended on average for experiment two, which in turn decreases precision as more visited spots are recommended, but also, compared to experiment two, more non-visited spots are recommended.

Feature group two shows in both cases some improvement for average precision, improving the average 1.51 percentage points for experiment one and 1.79 percentage points for experiment two. Its inclusion leads to a decrease in average recall of -0.95% for experiment one and -1.91% for experiment two.

A similar observation can be made for feature group three. The difference of average improvement in precision is more stronger between the experiments, being 1.42% for experiment one and 3.89% for experiment two. The change in recall is for both experiments negative, with -2.53% for experiment one and -1.88% for experiment two. Taking the average number of nodes as a proxy for the richness of the environment's description, we see a strong difference between the two experiments. For feature group two, the average number of spots in the vicinity is similar with 3.8 for experiment one and 3.72 for experiment two. For feature group three, experiment one has on average 2.52 type descriptions in the vicinity, compared to the much higher number of 8.47 type descriptions for experiment two. The differences for the two feature groups offers an explanation why they impact differently for the two experiments. Taking the average number of type descriptions as a proxy, feature group three

Table 1: Baseline results

	Experiment 1				Experiment 2			
	Naive Bayes		Ripper		Naive Bayes		Ripper	
	precision	recall	precision	recall	precision	recall	precision	recall
All spots	0.079	0.367	0.067	0.153	0.102	0.440	0.067	0.170
New Spots	0.045	0.243	0.037	0.071	0.050	0.276	0.046	0.110

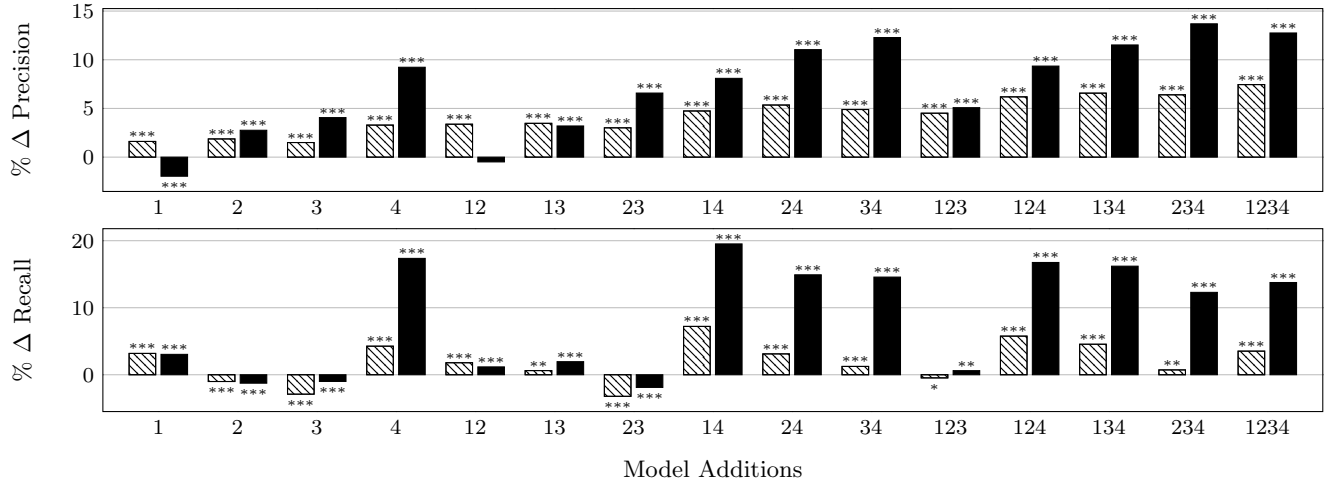

Figure 1: Changes relative to baseline model for precision and recall for all feature group combinations of experiment one and experiment two using Naive Bayes as classifier.

offers a much richer description of a spot's environment for experiment two than for experiment one than feature group two does, enabling the classifier to capture the description more precisely.

Lastly, feature group four shows an increase in both metrics for both experiments. Comparing the two experiments, we now see a strong difference between them. While the average improvement for experiment one is 3.19% for precision and 4.04% for recall, the average improvements are much stronger for experiment two, with on average 8.59% precision and 15.34% recall. This indicates that spots with more additional features result in a better precision and a higher recall.

Next, we take a look at the same models only for new spots the user did not visit before, i.e. those that were not used to learn his model. In Table 1, we see that the baseline model already has lower precision and recall for both experiments, which is not surprising, given that the spots are new to the recommender and thus harder to classify correctly.

The improvements induced by different feature group additions are shown in Figure 2. Comparing these values with the ones for all check-ins, one can observe that more models have a decreased average precision, especially for experiment two. For average recall, we observe a decrease for most models, sometimes up to 20 percentage points, while only a few of them showing a positive change. An observation is that feature groups two and three are always included in models with decreased average precision and average recall, which is especially true for experiment two. Thus, these feature groups seem to harm performance if new spots are to be recommended.

We again compare the averages of all models with a feature group with those that do not use a feature group. For new check-ins, feature group one has a positive impact on the average change in precision and recall in both experiments. While for experiment one, its inclusion on average increases precision 3.49% and recall 2.13%, for experiment two, precision and recall are increased 1.89% and 1.37%. This indicates that for new spots, the behavior of friends play a more important role than for spots already visited.

The average changes in precision and recall of feature groups two and three are negative for both experiments. For experiment one, feature group two decreases precision 1.37% and recall 5.05%. For experiment two, the drops are even higher, with 5.89% and 8.2%. For feature group three, average precision and recall for experiment one are decreased 1.21% and 8.37%. As the decrease in precision remains comparable to feature group two, the decrease in recall is stronger. For experiment two, the decrease in precision is with 4% not as strong as for feature group two, but the drop in average recall is with 14.6% more stronger. These results generally indicate that the inclusion of the feature groups lead to less spots being recommended, and compared to the baseline model, more often visited spots were not recommended.

Lastly, feature group four again shows on average positive effects for both experiments one and two. The impact is generally higher for experiment two, where the average increase of precision is 10.48% and recall is 8.01%, compared to experiment one, where average improvement is 1.38% in precision and 2.32% in recall.

In summary, for new spots, one would use feature groups one and four, leaving out groups two and three. The results seem to indicate that the environment features seem to perform poorly when confronted with new spots, a clue that a learner using them is not able to find generalized pat-

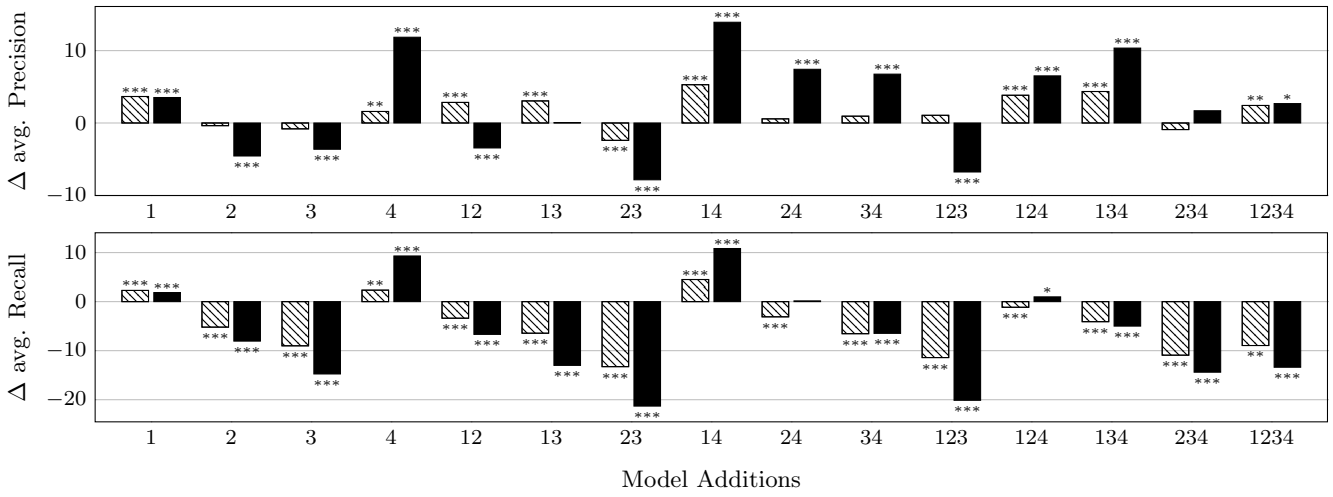

Figure 2: Changes relative to baseline model for precision and recall for new spots and Naive Bayes for experiment one and two.

terns in the environment information. This results in less visited spots being recommended that were recommended by the baseline model. 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.

When using Ripper instead of Naive Bayes for building user models, the results differ significantly. For both experiments, it shows a lower average precision and recall of the baseline model than Naive Bayes. Comparing the two experiments, one can see only a significant difference in average recall, while precision does not differ significantly.

Figure 3 shows the percentaged changes of individual models compared to the baseline model. Compared to the baseline model, nearly all models show a decline in average precision. Only model 02 in experiment one, i.e. including the Gowalla environment, can increase precision significantly. Average recall shows only improvements for model 02 and 024 in experiment one. On the other hand, some improvements are possible for the recommendation of *new* spots, as shown in Figure 4.

In summary, we only see minor improvements for some models from Ripper. The metric values for the improved model are still much lower for Ripper than for Naive Bayes. Additionally, one cannot observe a clear trend or stable influence of different feature groups, as the addition of a group to one model may increase a metric, while it decreases the same metric when added to another model.

7.2 Contextual Recommendation

The two contextual extensions we use are weather and time. Both can be included by applying them to the results of the recommendation discussed above, changing the lists of POIs to be recommended. This extension can be seen as an additional feature group, as post-filtering can be applied to any model from content-based recommendation.

Adding the context of weather did not improve recommendation for any model in our analysis, as optimal results using the test data were achieved when setting both thresholds to 1.0. Thus, the addition of weather data does not provide any knowledge which may improve the recommendation. In our approach, we put together data from different climate zones, which alone have very different characteristics. Although only taking data from North America, even there, a variety of climate zones exist, which can be seen by the example of cities with hight number of check-ins, such as San Francisco, Austin and New York, which have very distinctive weather patterns. 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 an approach promising for further work would be to segment the feature space created by weather information, with different models applying for different regions.

Adding the context of time on the other hand, we observe a significant improvement in average precision. The thresholds for different models are surprisingly similar, with a downgrade threshold for Naive Bayes between 0.73 and 0.75 and an upgrade threshold between 0.89 and 0.9. For Ripper, the optimal threshold for downgrade is between 0.76 and 0.77, while the optimal upgrade threshold is 0.66. Because of this, we chose to use the average threshold values and apply it to every model, to ease comparison.

The changes when contextual post-filtering is applied to models with different feature groups is generally similar. The average changes in precision and recall for both learners and both experiments can be seen in Table 2. Naive Bayes shows on average only small changes for average precision. While for experiment one, improvement is a bit more than

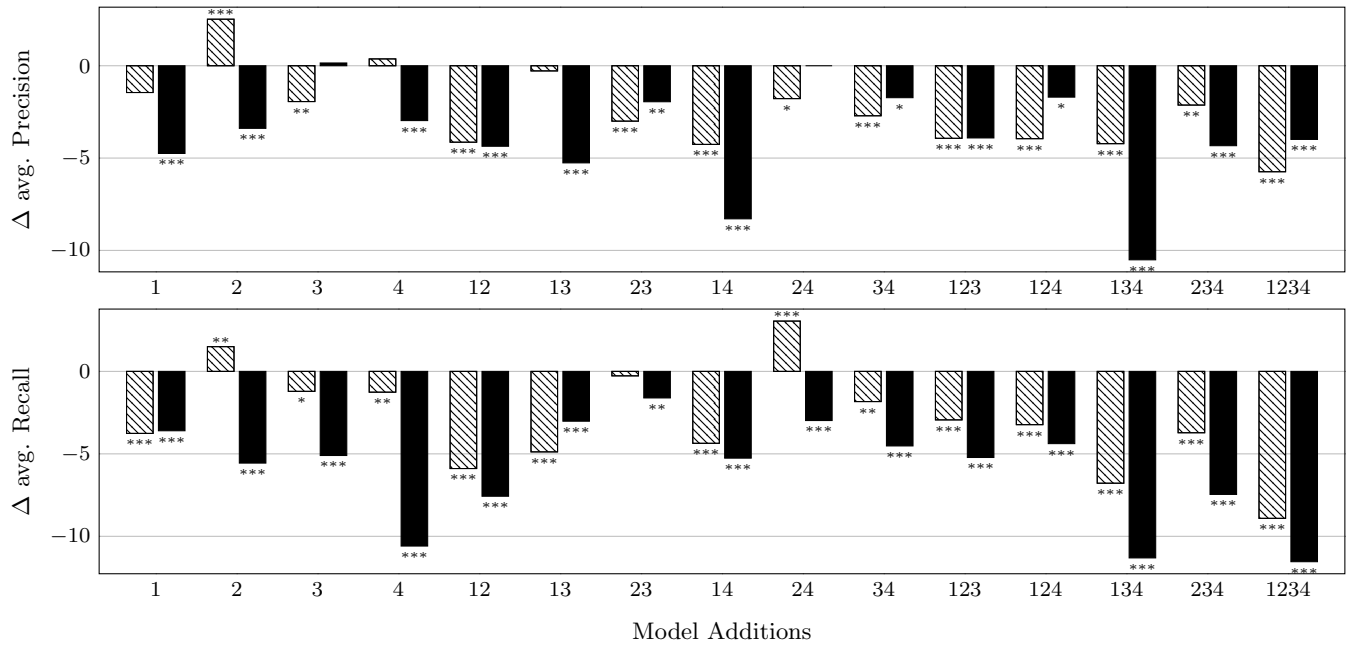

Figure 3: Changes relative to baseline model for precision and recall for all spots and Ripper for experiment one and two.

one percent, for experiment two, the average change is small and negative. Average recall is dropping more strongly, but still on average only between one to two percentage points.

The average improvement achieved for Ripper are much stronger for average precision and recall. With these improvements, models using Ripper and contextual recommendation become comparable to those of Naive Bayes, at least for experiment one. The best model, Ripper with feature groups 02 has now an average precision of 0.077, compared to 0.079 for the baseline model of Naive Bayes, which is the worst one in terms of precision. Average recall remains significantly different with 027 for Ripper and 0.37 for Naive Bayes.

Table 2: Average change in precision and recall for both classifiers and experiments, both for all and for new spots

Classifier	Exp.	Spots	precision	recall
NaiveBayes	One	all	1.41%	-1.88%
	One	new	1.07%	-1.25%
	Two	all	-0.72%	-1.85%
	Two	new	0.45%	-1.50%
Ripper	One	all	14.29%	72.61%
	One	new	47.85%	142.63%
	Two	all	8.25%	76.40%
	Two	all	19.76%	94.05%

When recommending new spots only, the average change of Naive Bayes are even smaller than for all spots. For Ripper, on the other hand, the changes are even stronger. Average precision is increasing nearly 48 percentage points for experiment one, while recall improves nearly 143 percentage points.

With these improvements, models using Ripper not only surpass the worst models from Naive Bayes, but become competitive to Naive Bayes when recommending new spots. For experiment one, the best Naive Bayes model, 04, has an average precision of 0.047 and a recall of 0.25. The best model for Ripper, (model 024 with context recommendation), average precision is significantly higher with 0.56, but at a significantly lower recall of 0.18.

For experiment two, Ripper coupled with a time-based, post-filtering extension not only surpasses the worst model of Naive Bayes, but even its best. When using Ripper in its best configuration (model 034 with post-filtering), one gets a precision of 0.055 and a recall of 0.211. Contrary to this, the best model for Naive Bayes, model 0234, has a average precision of 0.042 and an average recall of 0.16.

7.3 Combining LGD Features and Contextual Recommendation

Table 3 depicts the configurations for both learners with contextual information. It can be observed that the precision and recall for Naive Bayes can be improved by 14.34% and 19.52%, respectively, while the relative improvement for Ripper is larger (51.08% and 150.08%, respectively), but starting from a lower baseline model and not reaching the absolute values achieved with Naive Bayes.

In summary, this shows that given the selection of good features and a working post-filtering algorithm, the results even of weaker base classifiers can be strongly improved.

8. CONCLUSION

In this paper, we have shown an approach for enhancing POI recommendations for a location-based social network by enriching it with semantic data. For this, we used a dataset of POIs and check-ins crawled from Gowalla[10], added knowl-

[10]Even though Gowalla has been shut down in the meantime,

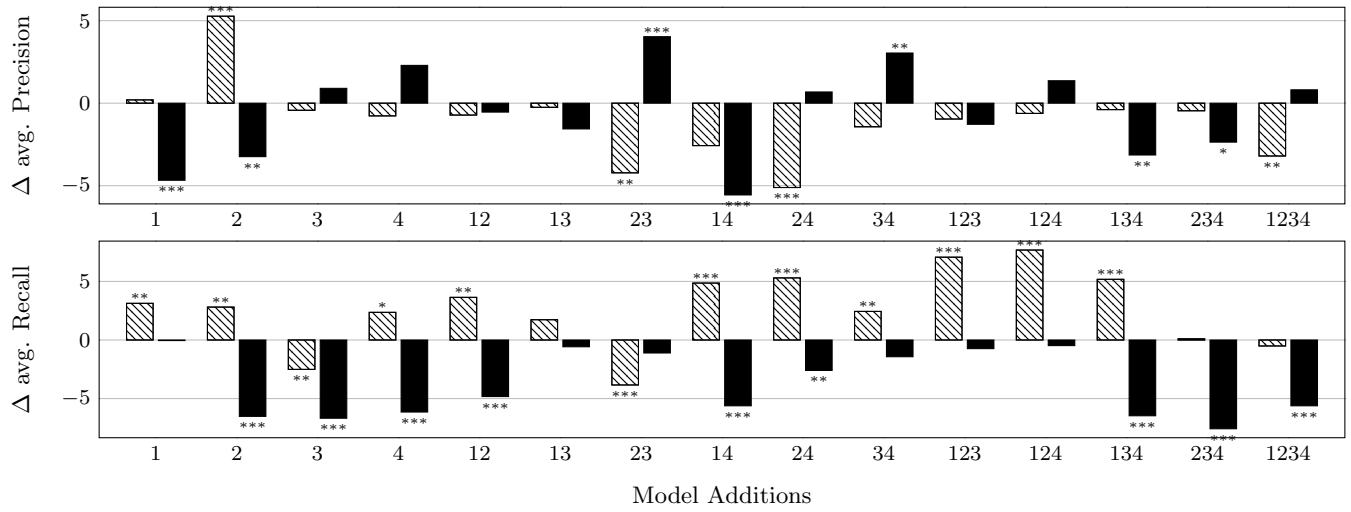

Figure 4: Changes relative to baseline model for precision and recall for new spots and Ripper for experiment one ▨ and two ▮.

Table 3: Comparison of best models with respect to precision and recall with and without contextual recommendation. The numbers in parantheses denote the relative improvement w.r.t. to the baseline model of the respective learner.

exp.	spot	learner	Model	context	prec.	recall
1	all	NB	01234	yes	**0.087 (9.04%)**	0.372 (1.60%)
1	all	NB	04	no	0.083 (4.74%)	**0.393 (7.21%)**
2	all	NB	0234	no	**0.117 (14.34%)**	0.485 (10.41%)
2	all	NB	014	no	0.110 (8.09%)	**0.525 (19.52%)**
1	new	NB	014	yes	**0.029 (6.18%)**	0.25 (2.90%)
1	new	NB	014	no	0.029 (5.29%)	**0.19 (4.51%)**
2	new	NB	014	no	**0.056 (13.92%)**	**0.306 (10.83%)**
1	all	Ripper	02	yes	**0.077 (15.48%)**	0.266 (74.17%)
1	all	Ripper	024	yes	0.076 (14.25%)	**0.266 (74.76%)**
2	all	Ripper	0234	yes	**0.073 (8.97%)**	0.296 (74.29%)
2	all	Ripper	0	yes	0.071 (6.31%)	**0.298 (75.42%)**
1	new	Ripper	02	yes	**0.056 (51.08%)**	0.174 (145.42%)
1	new	Ripper	0123	yes	0.054 47.20%	**0.177 (150.08%)**
2	new	Ripper	023	yes	**0.057 (23.07%)**	0.212 (93.00%)
2	new	Ripper	01	yes	0.054 (17.79%)	**0.213 (93.96%)**

edge about the POIs, taken from LinkedGeoData (LGD), a Linked Data version of Open Street Map, and the context of a check-in in the form of weather information, taken from MesoWest, a service offering weather information from North America.

In the first case, we first linked POIs from Gowalla with the corresponding ones from LGD, utilizing properties such as the name, location and category. Then, we generated features from the linked entities using FeGeLOD, a tool that allows to create features using reasoning heuristics. Additionally, we created features by using the distribution of the POI's categories in the vicinity of a spot. For weather data, we created a list of check-ins nearby weather stations, then crawled the weather information form MesoWest.

Using this additional knowledge, we created additional feature groups, serving as the basis for content-based recommendation. Apart from knowledge derived from linked POI itself and the environment description, we also generated an environment description of nearby spots in Gowalla and a group of friend-based features. Apart from the context information from MesoWest, we also used the time of check-ins as context-based features.

With these feature groups, we evaluated recommendation performance, offline experiments for content-based experiments, systematically varying different combination of feature groups, learning algorithms, user and check-in groups. To add contextual information, we used a post-filtering approach, where the result of the content-based recommendations are re-weighted depending on the context.

The results show that some combinations of feature groups can improve recommendation results. In particular, additional attributes of the spots, obtained from LinkedGeoData, work well, as well as social features. Furthermore, it showed that using an appropriate amount of context data

the approach can be transferred to other LSNs in a straight forward way.

can even level out significant differences between base learners.

In summary, the results show both that the intelligent use of existing knowledge in databases, but also the addition of external knowledge can enhance recommendation. For the first case, we see potential in utilizing the current position of a user's friend for recommendation. The literature shows that the user's behavior is strongly affected by his peers, thus recommending spots that are for example currently visited by one's friend seems to be an approach worth evaluating. Additionally, the knowledge of the previous locations of users can be used to enhance recommendation, as it has been shown that patterns regarding the transition from different kind of POIs exist. This information can be derived directly from the information available, making it unnecessary to add information.

On the other hand, external knowledge can greatly enhance knowledge about the entities involved in a recommendation and thus improve it. Especially Linked Data may be of great benefit, as on its basis the inference of new, implicit knowledge is possible, as we have shown in this article. While we only used data from one dataset, the results show the clear potential of the approach, and future work should look at combining background knowledge from different datasets.

In summary, this paper has shown that background knowledge from structured data sources on the web clearly adds value when building recommender systems.

9. REFERENCES

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145, 2005.

[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] S. Auer, J. Lehmann, and S. Hellmann. LinkedGeoData: Adding a Spatial Dimension to the Web of Data. In A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, editors, *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 731–746. Springer Berlin Heidelberg, 2009.

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

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

[6] L. Baltrunas and F. 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.

[7] B. Berjani and T. 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.

[8] Betim Berjani. Recommendation Systems for location-based Online Social Networks. Master's thesis, Technische Universität Darmstadt, Darmstadt, 2010-12-14.

[9] W. W. Cohen, P. D. Ravikumar, and S. E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In S. Kambhampati and C. 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.

[10] G. Gupta and W.-C. Lee. Collaborative Spatial Object Recommendation in Location Based Services. In *2010 39th International Conference on Parallel Processing Workshops*, pages 24–33. IEEE, 2010.

[11] 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.

[12] 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.

[13] Mohamed, M. Ye, P. Yin, and W.-C. 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.

[14] 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.

[15] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. 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.

[16] H. Paulheim and J. Fürnkranz. Unsupervised Generation of Data Mining Features from Linked Open Data. In *International Conference on Web Intelligence and Semantics (WIMS'12)*, 2012.

[17] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer US, Boston and MA, 2011.

[18] M. Setten, S. Pokraev, and J. Koolwaaij. Context-Aware Recommendations in the Mobile Tourist Application COMPASS. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, P. M. E. Bra, and W. Nejdl, editors, *Lecture Notes in Computer Science*, pages 235–244. Springer Berlin Heidelberg, Berlin and Heidelberg, 2004.

[19] C. Stadler, J. Lehmann, K. Höffner, and S. Auer. LinkedGeoData: A core for a web of spatial open data. *Semantic Web*, 3(4):333–354, 2012.