Driver Recommendations of POIs using a Semantic Content-based Approach

Rahul Parundekar and Kentaro Oguchi

Toyota InfoTechnology Center, U.S.A.
Mountain View, CA
{rparundekar,koguchi}@us.toyota-itc.com

Abstract. In this paper, we present a semantic content-based approach that is employed to study driver preferences for Points of Interest (POIs), e.g. banks, grocery stores, etc., and provide recommendations for new POIs. Initially, logs about the places that the driver visits are collected from the cloud-connected navigation application running in the car. Data about the visited places is gathered from multiple sources and represented semantically in RDF by 'lifting' it. This semantic data is then combined with driver context and then input into a machine learning algorithm that produces a probabilistic model of the driver's preferences of POIs. When the driver searches for POIs in an unknown area, this preference model is used to recommend places that he is most likely to prefer using a nearest-neighbor approach. In this paper, we describe the details of this content-based approach for recommendation, along with the results of a user study that was conducted to evaluate the approach.

Keywords: Semantic Web, Recommendations, Preference Modeling

1 Introduction

In deploying connected services to the car, it is important that driver safety is given a high priority. To reduce possible distractions in accessing information, the in-vehicle navigation system enforces restrictions on the way information is presented to the driver. One such constraint is that the number of items that can be displayed in a list on a single screen is fixed to a limited number of slots. When the driver searches for banks in the car, for example, the search results are displayed as a list filled in the available slots. If the number of search results exceeds the number of slots, then the extra results are pushed to the next page. Accordingly, there arises a need for presenting the most relevant information to the driver in those slots. We model this, as a recommendation problem of providing personalized, contextualized information to the driver. In particular, this paper discusses the recommendation of Points of Interest (POIs), e.g. banks, grocery stores, etc., in the car using a combination of semantic technologies and a recommendation system.

In our previous paper on Learning Driver Preferences of POIs using a Semantic Web Knowledge System[8], we presented the architecture and components of a semantic system that is able to model the driver's preferences. In this paper, we present details of the recommendation aspect of the work including the reasons for the selection of a content-based approach, the details of the nearest-neighbor recommendation method and the results of a user study that was conducted earlier this year. Using the history of Points of Interests (POIs) that the driver visits, we can build a model of the places he/she (henceforth referred to as *he*) is more likely to prefer. For example, the driver may have certain preferences for banks. Based on the ones he has visited in the past, we try to build a preference model that can be used to determine his affinity for a bank he has not visited before. The next time he is searching for a bank in an unfamiliar place, his preference model is used to recommend him one from the banks around him.

The paper is organized as follows. We first provide a detailed explanation of the recommendation task and describe the reasons for the selection of a content-based approach. We then describe how we learn the driver preferences. This is followed by an explanation of how the learned model is used in the recommendation of POIs. We also briefly describe the underlying system, which uses semantic technologies in the data representation and services, called the Semantic User Preference Engine or *Supe*. This is followed by a description of the user study that was conducted using an implementation of Supe, and its results. We also include relevant work in the Semantic Web and recommendation literature, for modeling user preferences and recommendations. Lastly, we conclude by summarizing our findings along with future work.

2 Using a Content-based Approach for Recommendation

The selection of the algorithm for POI recommendation in the car is largely based on two issues: the nature of the data available and desired recommendation to be produced. In the first case, data comes from the usage history of the navigation application in the vehicle. The user can select a place to navigate to in three ways, as can be seen in Fig. 1: (1.a) driver chooses a POI from the head-unit; (1.b) user selects a POI from a suite of installed applications on his smartphone or (1.c) user pre-selects a POI from his desktop/browser application and 'sends' the POI over the Cloud to the navigation application running in the car. The selected POIs from these connected devices can be tracked on the Server running in the Cloud before sending it to the navigation application. As we can only track places that the driver has visited, which we assume he likes, we only have positive training examples. For the second issue, we want to try to provide a recommendation that answers the question, 'Which POIs among the ones available around the driver is he most likely to prefer?', when the user searches for POIs. Accordingly, the recommender system should be able to choose the place most likely to be preferred by the user from a set of candidate places returned by the database.

Recommender systems are popularly classified into *Collaborative Filtering*, *Content-based* approaches or a mixture thereof[2, 1]. Common issues with the *Collaborative Filtering* algorithms are the new user problem, new item problem and the sparsity problem. Out of the three, data sparsity is the biggest possible

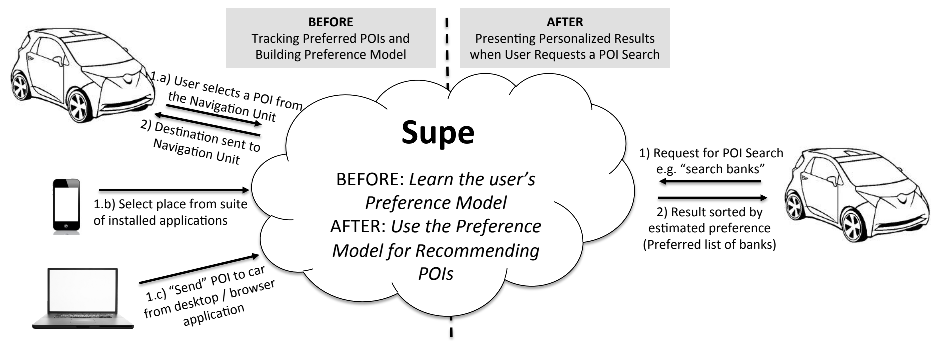

BEFORE
Tracking Preferred POIs and Building Preference Model
AFTER
Presenting Personalized Results when User Requests a POI Search
1.a) User selects a POI from the Navigation Unit
2) Destination sent to Navigation Unit
1.b) Select place from suite of installed applications
1.c) "Send" POI to car from desktop / browser application
Supe
BEFORE: Learn the user's Preference Model
AFTER: Use the Preference Model for Recommending POIs
1) Request for POI Search e.g. "search banks"
2) Result sorted by estimated preference (Preferred list of banks)

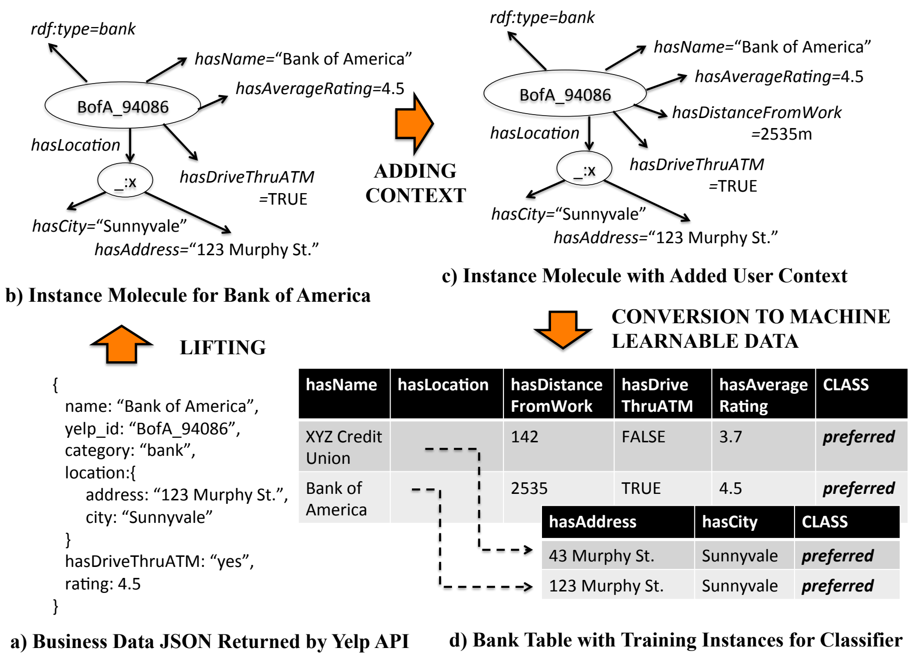
Fig. 1. Recommendation Task Overview

reason of failure of recommendation in the dataset. As an example, consider a person who lives in San Jose, CA. The places that he visits frequently in his neighborhood - e.g. his bank, grocery store, etc., along with history of others in his neighborhood become part of the training dataset. When the user travels to a remote place - e.g. Livermore, CA (about 40 miles away) and wants to search for a bank, it is highly unlikely that there would be another set of users who go to the same bank as the user in San Jose *and* might have visited a bank in that particular neighborhood in Livermore. Contextual information is also important in providing relevant recommendations. Though *Collaborative Filtering* techniques that include context by introducing new dimensions apart from the traditional two - *Users* and *Items* - have been studied[2], they suffer from worse data sparsity problems despite optimizations for computational overhead.

In contrast, a *content-based* approach seems more intuitive. In the first example, the user might prefer a certain banking chain, which has a branch in his neighborhood and he would prefer a local branch when he is searching for ATMs in Oakland. In case of gas stations, he may have a preference for cheaper gasoline. A single *Collaborative Filtering* approach would be insufficient for recommending such cases, especially in the case of fluctuating gas prices. Contextual information can be appended to the POI data as extended attributes to generate a context-specific version of the item, which can then be used to provide recommendations. The content-based approach for POI recommendation employed in this paper, is similar to other recommender systems in relevant literature [11, 6] and is described in the following sections.

3 Generating the Preference Model From Driver History

Before we can recommend POIs to the driver, we first need to generate a model of his preferences. To do so, the data collected from his navigation history is first converted into machine learnable data. When connected devices send data to the in-vehicle navigation application or the driver selects a destination on the navigation system, Supe tracks the visited/consumed POIs and stores them as driver history. The process of generation of the user's preference model from this data is described below. Fig. 2 shows the steps involved in converting POI data about a bank, visited by the driver, into machine learnable data.


rdf:type=bank
hasName="Bank of America"
hasAverageRating=4.5
BofA_94086
hasLocation
_:x
hasDriveThruATM =TRUE
hasCity="Sunnyvale"
hasAddress="123 Murphy St."
b) Instance Molecule for Bank of America
ADDING CONTEXT
rdf:type=bank
hasName="Bank of America"
hasAverageRating=4.5
BofA_94086
hasDistanceFromWork =2535m
hasLocation
_:x
hasDriveThruATM =TRUE
hasCity="Sunnyvale"
hasAddress="123 Murphy St."
c) Instance Molecule with Added User Context
LIFTING
CONVERSION TO MACHINE LEARNABLE DATA
a) Business Data JSON Returned by Yelp API
d) Bank Table with Training Instances for Classifier

```
{
  name: "Bank of America",
  yelp_id: "BofA_94086",
  category: "bank",
  location:{
     address: "123 Murphy St.",
     city: "Sunnyvale"
  }
  hasDriveThruATM: "yes",
  rating: 4.5
}
```

hasName	hasLocation	hasDistance FromWork	hasDrive ThruATM	hasAverage Rating	CLASS
XYZ Credit Union		142	FALSE	3.7	***preferred***
Bank of America		2535	TRUE	4.5	***preferred***

hasAddress	hasCity	CLASS
43 Murphy St.	Sunnyvale	***preferred***
123 Murphy St.	Sunnyvale	***preferred***

Fig. 2. Building User Preferences (Note: Only a subset of the actual attributes used is shown.)

Fetching RDF data for the Visited POIs: Data for the POIs, which were collected as driver history, is retrieved from multiple sources like a POI database or Web Services (by searching for the POI using information like its name, location, address, etc.). For example, the JSON response from a Web Service (e.g. Yelp API) for the "Bank of America" branch, which the user visited, is shown in Fig. 2 (a). As each of these sources might have different schemas, it is necessary to integrate this information together into a common vocabulary. At the heart of Supe is a Places Ontology (see Fig. 3). The ontology defines a concept hierarchy of places along with the properties associated with each concept. Data from the different sources is 'lifted' into this ontology and merged together. For the "Bank of America" that the user visited, the lifting process converts the JSON response of the Web Service into an RDF instance molecule [4] as shown in Fig. 2 (b). Because of the simplicity of the POI domain, the lifting rules were determined at design time.

Adding Context: Data from the driving history along with the driver's personal information is then used to automatically generate context information. The relevant triples, generated using pre-defined rules, are then added to the RDF instance molecule from the previous step. For example, if the user has his home or work address stored, then the *distance from home* or *distance from work*

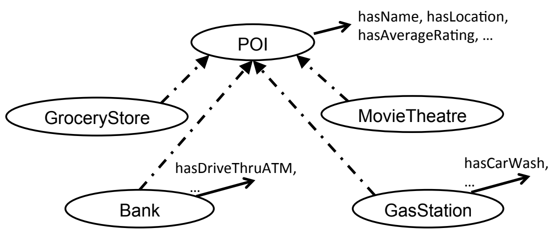


Fig. 3. Place Ontology (partial)

contexts can be added. User context *hasDistanceFromWork* is added to the instance molecule in Fig. 2 (c). The user's preference model can now be trained with the instance molecule generated.

Converting the RDF Instance Molecule to Machine Learnable data: Since we use a content-based approach for recommendation, we need to convert RDF into a representation that machine learning algorithms can understand. The translation of instance molecules into a table of training data, as used by conventional machine learning algorithms, is relatively straight-forward and is explained below. (Due to lack of space, Fig. 2 (d) only shows a representative set of columns that can be derived from Fig. 2 (c)).

1. **The Table:** All instances belonging to one concept are grouped together in a single table, e.g. banks in Fig. 2 (d).
2. **Rows in the Table:** Each instance molecule to be added to the preference model translates to a row in the table. Instead of dropping the URI, as identifiers usually do not contribute to the learned model, we add the identifier to the table (not depicted in Fig. 2 (d)) to bias the model with a higher preference to a previously visited place.
3. **Columns in the table** The attributes or property-value pairs for the instance get translated as columns and values in the table. The properties for which the type of the table is a domain, appear as columns. For example, while the *hasDriveThruATM* is a property of Banks, the *hasName* property is inherited from the parent concept *POI* and is also present in the table in Fig. 2.
4. **Values in the Table:** RDF Literals in a column are translated as one of string, numeric or nominal values. The relevant information for the conversion into either of these types can be determined from the ranges of the properties in the ontology, or specified explicitly during ontology construction. For example, a value of the *hasName* property translates to string type, the *hasAverageRating* property translates to numeric type and the *hasDriveThruATM* property translates to nominal type. For values of properties that are RDF Blank Nodes, we use nesting of tables where we track inner values for the properties of the blank node (e.g. *hasLocation* property) . For properties with URI values, we can choose to either use the lexical value of the identifier as cell value in the table if we want to introduce bias, or represent the instance in a nested table similar to blank nodes if the values

of its properties are important. For missing attributes, the cells in the table are empty.

5. **Class Column in the Table:** Since visited places translate to only positive training examples, all class values in the table are marked as 'preferred'

Building the Preference Model: The table above, similar to the training data used for a naïve-Bayes classifier, is then used to generate a preference model. The preference model generated is an optimized variation of the table, containing frequency counts/mathematical functions, and helps in calculating the likelihood probabilities. Instead of building the entire preference model from scratch every time a new instance is added, we use an incremental approach.

4 Recommending POIs using a *content-based* approach

In *content-based* recommendation systems, determining if the user likes/dislikes a particular item is a classification problem [11]. A variety of algorithms, like linear regression, rule-based, probabilistic methods, etc. can be used for determining items for recommendation. For example, a naïve-Bayes approach can be used to build a model that can classify a previously unseen POI as *preferred* or *not-preferred*. One can determine which one of n candidate POIs is most likely to be preferred by the user by selecting the POI with the highest (normalized) probability of it being preferred ($P(\mathit{preferred}|\mathit{POI}_i)$). This item can then be recommended to the user. Since the driver history that was collected only contains positive training examples, most of the algorithms mentioned above cannot be used as-is (e.g. in a naïve-Bayes approach, the likelihood and evidence probabilities cancel each other out, giving an inappropriate probability score). In the absence of negative training examples (i.e. places that the user dislikes), we use a nearest-neighbor approach for recommendation. Specifically, from n candidates, which may have been previously unseen, we recommend the item that best matches the ones in the user's preference model. Our approach is described below.

Fetching Candidate POIs and Adding Context: When the user wants to search for POIs, Supe first retrieves candidate places that match the search criteria from the POI Database/Web Services. Necessary user and situation context are added to the POIs after lifting them into RDF, similar to the steps described in Section 3. These POIs can then be scored to find out how likely is the user to prefer each of them.

Nearest-neighbor to a hypothetical Bliss-point Our approach is inspired from the nearest-neighbor algorithms used in clustering and classification.To find how likely is the user to prefer each POI, we first find out how likely is the user to prefer its properties. Let's suppose that each candidate POI has only two properties - name and average rating of all users. The first candidate POI *hasName* "Bank of America" and *hasAverageRating* of 3.5. We can calculate the likelihood probabilities - $P(hasName = \text{``}BankofAmerica\text{''}|preferred)$ and $P(\mathit{hasAverageRating} = 3.5|preferred)$ using the table from Fig. 2, similar to the likelihood calculation in a naïve-Bayes approach. We can plot this point in a Euclidean space with the properties as the different axes (see Fig. 4 (a)). For a hypothetical POI that is *always* preferred by the user, each of its attributes would have the preference likelihood as 1.0. We can now plot this 'bliss-point' on the Euclidean space. We can similarly plot other candidate POIs (see Fig. 4 (b)). The euclidean distance from the bliss-point is used for the recommendation, and the POI that is nearest to the bliss-point is recommended to the user.

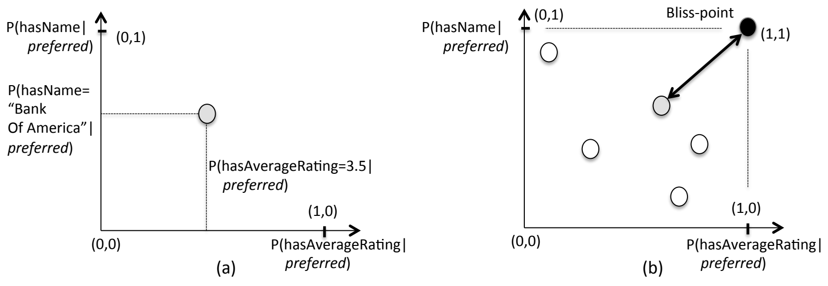

P(hasName| preferred)
(0,1)
P(hasName= "Bank Of America"| preferred)
P(hasAverageRating=3.5| preferred)
(1,0)
(0,0)
P(hasAverageRating| preferred)
(a)
(0,1)
Bliss-point
(1,1)
P(hasName| preferred)
(1,0)
(0,0)
P(hasAverageRating| preferred)
(b)

Fig. 4. Plotting candidate POIs using the likelihood probabilities of their attributes. (a) Plotting one candidate. (b) Recommending the nearest neighbor to the Bliss-point.

Distance from Bliss-point We can easily extend the above method to an n-dimensional space. The values of the attributes in the columns in the table in Fig. 2 (d) can either be Literals, Blank Nodes, URIs or missing. For literal values, the likelihood probability is calculated as follows: (i) a Gaussian distribution is used for the probabilities of numeric values (ii) a document similarity metric is used for the probabilities of string values and (iii) symbol probability is used for the probabilities nominal values. The distance for a property with a blank node is calculated recursively, by first calculating the likelihood probabilities of its inner values, and then its distance from another hypothetical bliss-point in its own high dimensional space. For URIs, we can use a dual strategy depending on the nature of the property. In some cases, to bias toward previously seen values for properties, we calculate likelihood as its probability of the occurrence of that URI in that column. If the attributes of the corresponding instance are more important, then the likelihood can be calculated similar to the blank node. The distances from the bliss-point of multiple POIsneed to be normalized before they can be compared. This is done by dividing the distance by the distance of the origin to the bliss-point, thus taking care of missing attributes. For a multivalued property, we take the average of the distances of all its values. As an example, the distance from bliss-point for the Bank of America POI in Fig. 2 (d), would be calculated as follows.

$$D(BofA\_94086) = \sqrt{\frac{\begin{array}{r}(1 - P(\text{``}Bank\ of\ America\text{''} \mid preferred))^2 \\ +(1 - P(2353 \mid preferred))^2 + (1 - P(TRUE \mid preferred))^2 \\ +(1 - P(4.5 \mid preferred))^2 + D(\_ : x)^2\end{array}}{5}}$$

$$Where\ D(\_ : x) = \sqrt{\frac{(1 - P(\text{“}Sunnyvale\text{”} \mid preferred))^2 + (1 - P(\text{“}123\ Murphy\ St.\text{”} \mid preferred))^2}{2}}$$

Recommending the POIs to the User The candidate items retrieved from the database are scored using the *distance from bliss-point* metric and sorted according to the distance. The sorted list of POIs is then sent by the Supe system to the navigation application as a recommendation. Once the user selects a POI from the list, it is fed back to the preference model and the system is able to learn incrementally.

5 The Supe Semantic Web Knowledge System

Since the Supe system is described in detail elsewhere [8], we only describe an overview (see Fig. 5). Supe is a Semantic Web Knowledge System used to collect driver preferences and apply the preference model to provide personalized POI search results to the driver. It is based in the Cloud and contains a Knowledge Base, Intelligent Services, RESTful endpoints and access control mechanisms. To be successful in modeling driver preferences, it needs driver as well as POI data. This semantic data is grounded in an ontology and represented as Linked Data in the Knowledge Base. Supe also provides Intelligent Services, associated with the machine learning task described in the previous sections, for learning the driver's preferences and finding the recommended POIs for the driver. These are wrapped by thin RESTful services that are accessible to the navigation application running on the head-unit and other connected devices for searching POIs and pushing POIs to the navigation application via the Cloud. To prevent RESTful services, Intelligent Services and applications that are running on the connected devices from corrupting other services' or users' data, Supe has an access control mechanism in place. Applications and users use an identifier and a secret passkey combination for authentication. The authorization mechanism is tied to a hierarchical namespace scheme for URIs that governs the policies for data ownership. The RESTful services and Linked Data (URIs starting with 'https://') are accessible on the connected devices side using secured (HTTPS) communication, thus ensuring confidentiality.

6 User Study & Evaluation

We implemented a smart-phone application emulating the in-vehicle navigation application for a user study. The application allowed users to search for POIs belonging to different categories (e.g. gas stations, banks, restaurants, etc.). A cloud-based server supporting the application was implemented using the description of Supe above. Around 50 people, who used their car for running household chores or for their daily commute to work, in the Bay Area (around San Francisco, CA) were selected for the user study. Each user was asked to add at least 10 places to his preference model by driving to the POIs he would visit in

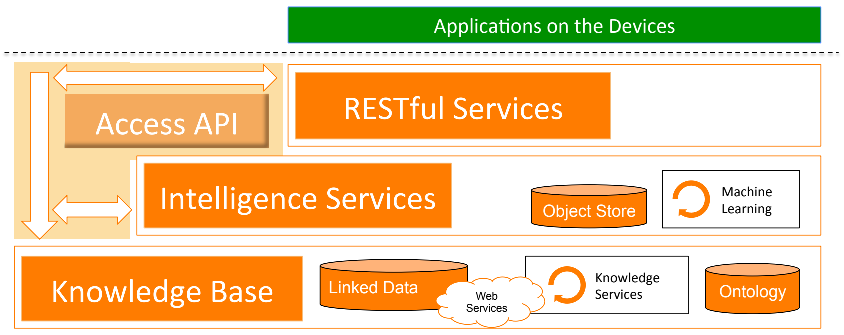
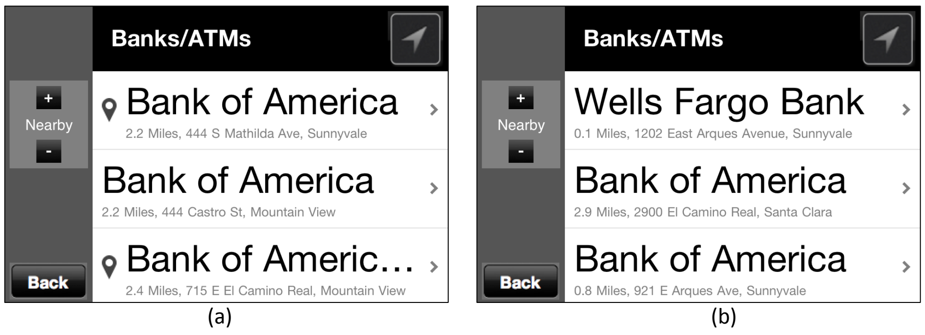


Fig. 5. Semantic Web Knowledge System Layer Cake

his daily life and letting the application know that he like the place, by clicking a 'like' button. Each user was also asked to perform 10 search tasks, where he would search for POIs and choose to one out of the recommended places. The application usage was recorded.

The following screenshot (Fig. 6) shows two instances of POI search for one such user. Initially, there are 3 slots for displaying recommendations, and any other places in the recommended POI list are scrollable. The first image shows the results for search for banks in the user's daily commute area. The ones marked with a pin are the places that he has visited (and liked) and are used to build his preference model. When the user was in a remote place and in need of cash, he searched for banks using the application. The second capture shows that two of his preferred banks were ranked among the top three in the list. Intuitively the reader may figure out that this user prefers a specific banking company. The model is able to detect this preference because its likelihood probability on the *hasName* axis is high. Similar patterns were also detected for other users in different categories, like gas stations, restaurants, etc.



Bank of America
2.2 Miles, 444 S Mathilda Ave, Sunnyvale
Bank of America
2.2 Miles, 444 Castro St, Mountain View
Bank of Americ…
2.4 Miles, 715 E El Camino Real, Mountain View

Wells Fargo Bank
0.1 Miles, 1202 East Arques Avenue, Sunnyvale
Bank of America
2.9 Miles, 2900 El Camino Real, Santa Clara
Bank of America
0.8 Miles, 921 E Arques Ave, Sunnyvale

Fig. 6. Screenshots of the Implemented User Study Application: Search result for banks (a) in daily commute area (visited places marked) (b) in an unfamiliar area (using distance from bliss-point metric).

The users had been given two weeks to complete the tasks during which observations were made on the performance of their preference model. For each POI search task, a successful recommendation was counted if, from the 20 candidate POIs that the database returned, the user selected one of the top 3 recommendations. We tracked the success-rate (percentage of successful recommendations in the search task) of the preference model with the increasing size of the preference model and also with the number of searches performed. The selected POI was fed back to the preference model for learning the preference incrementally. Fig. 7 shows the performance of the preference model for one user. As can be seen, after the initial stabilization period, the success rate steadily improved with increasing number of instances in the preference model and as more searches were performed. Overall, for the 48 participants that completed the task, the success rate at the completion of the tasks was 47.63% on an average. This is better than a strategy for random recommendation of 3 out of 20 candidate POIs, which would result in a 15% success rate. Though the improvement in the success-rate was satisfactory, there is a clear scope for better performance. In the future, we intend to use this result as a baseline for comparisons with updates to the learning technique.

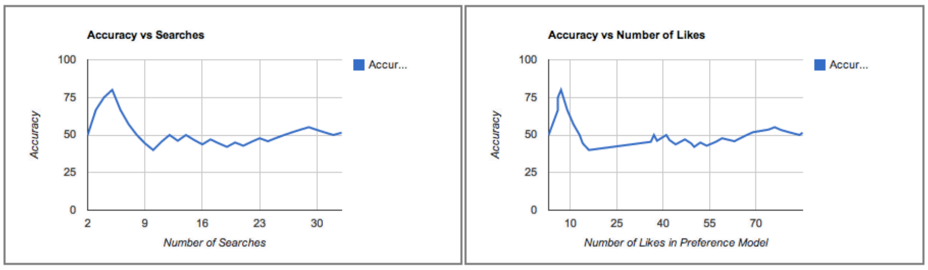


Fig. 7. Performance of the preference model for recommendations made to one user

The use of a probabilistic approach also allowed us to 'look inside' the preference model of a user. For example, for the user in Fig. 6, his preferences for the *hasName* attribute (word cloud) and *distance from current location* context (distribution) are shown in Fig. 8. These visualizations were useful for intuitively understanding the likelihood probabilities of the properties.

7 Related Work

Content-based preference modeling has received large research attention in recommendation systems over the past few years[11]. These algorithms, use machine learning techniques like linear regression, naïve-Bayes, etc. for finding recommendations. One of the earliest works, the Syskill & Webert system[10] uses a naïve-Bayes classifier for classifying web sites as either 'hot' or 'cold'. Similar to our approach, this system also uses the probability score to rank pages according to user's preferences. More recently, personalized information retrieval has also

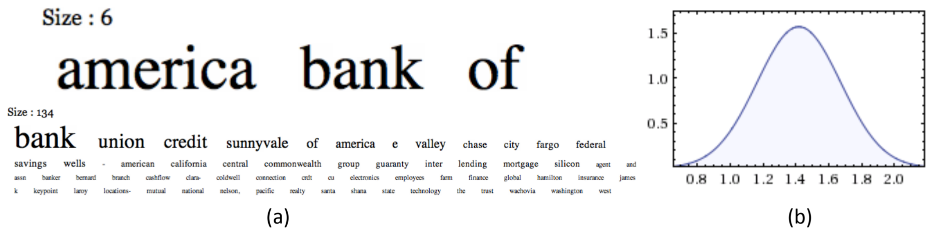


Fig. 8. Performance of the preference model for Bank recommendations. (a) Word cloud for the *hasName* property for banks, showing user's preference for a banking company over others. (b) Distribution of the values for the *hasDistance* property showing user's preference for banks averaging about 1.4 km away.

been gaining traction in the Semantic Web. For example, *dbrec*[9] is a music recommendation system based on DBpedia that uses a semantic link distance metric for its recommendations. An important constraint that restricted our use of more sophisticated algorithms was the absence of any negative training examples. Incidentally, one-class classification approaches have also been studied, which may present an interesting alternative to our method[12, 5]. In these, clustering, kernel based, or other methods are used to identify the boundary of the class and then predict if an item belongs to that class or is an outlier. The metric used for determining outliers can possibly be used as an alternative to the *distance from bliss-point* metric.

In the past year, the combination of RDF and machine learning has gained some traction. The work that is perhaps most similar to our approach on converting RDF to a machine learning table, is found in Lin et. al [6]. For the movie domain, they try to predict if a movie receives more than $2M in its opening week by converting the RDF graph for the movie to a Relational Bayesian Classifier. One major difference in their approach to ours is the translation of multi-valued object properties (e.g. *hasActor*) into the relational table. While doing so, their approach 'flattens' all objects. For example, names of all actors get aggregated into a single 'has actor name' column and all actors' year of birth get aggregated into a 'has actor YoB' column. The associativity within an instance (e.g. of an actor's name to his age) is lost. In our approach, this advantage, of property associativity using a graph, is maintained since we determine the score for each blank node independently. However, a more in-depth comparison of the two approaches needs to be conducted on common data for better analysis. Another related work in learning from RDF has been explored in Bicer et. al[3], where relational kernel machines are used for movie recommendations. A similar kernel based approach was also used in Lösch et. al [7] for recommending new links in RDF graphs - for example, recommending known people (*foaf:knows*).

8 Conclusion & Future Work

In this paper we described a semantic content-based approach for recommending POIs according to driver preferences learned using the navigation history.

Our approach was able to build the preference model using RDF data associated with the driver history as a result of the following: (i) it used an easy translation method between semantic content (RDF data) into machine learnable tables (ii) the content-based approach was easily extendable to include and learn the contextual preference (e.g. *distanceFromWork*, etc.) (iii) the *distance from bliss-point* metric was used to recommend POIs from a set of candidates using the modeled preferences (iv) the probabilistic nature of the likelihood of the attributes in the preference model helped study the user's preferences by visualization. This was verified through a user study that produced a 47.63% success-rate.

Though our preliminary evaluation shows promising results with actual users, using simple metrics, it has scope for improvements. We intend to explore other machine learning techniques (e.g. kernel based, one-class classification, etc.) to improve the preference model, as future work. We also intend to test this on a larger dataset of driver history.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on 17(6), 734–749 (2005)
2. Almazro, D., Shahatah, G., Albdulkarim, L., Kherees, M., Martinez, R., Nzoukou, W.: A survey paper on recommender systems. Arxiv preprint arXiv:1006.5278 (2010)
3. Bicer, V., Tran, T., Gossen, A.: Relational kernel machines for learning from graph-structured rdf data. The Semantic Web: Research and Applications pp. 47–62 (2011)
4. Ding, L., Finin, T., Peng, Y., Da Silva, P., McGuinness, D.: Tracking rdf graph provenance using rdf molecules. In: Proc. of the 4th International Semantic Web Conference (Poster) (2005)
5. Khan, S., Madden, M.: A survey of recent trends in one class classification. Artificial Intelligence and Cognitive Science pp. 188–197 (2010)
6. Lin, H., Koul, N., Honavar, V.: Learning relational bayesian classifiers from rdf data. The Semantic Web–ISWC 2011 pp. 389–404 (2011)
7. Lösch, U., Bloehdorn, S., Rettinger, A.: Graph kernels for rdf data. The Semantic Web: Research and Applications pp. 134–148 (2012)
8. Parundekar, R., Oguchi, K.: Learning driver preferences of pois using a semantic web knowledge system. The Semantic Web: Research and Applications pp. 703–717 (2012)
9. Passant, A.: Dbrec, music recommendations using dbpedia. The Semantic Web–ISWC 2010 pp. 209–224 (2010)
10. Pazzani, M., Billsus, D.: Learning and revising user profiles: The identification of interesting web sites. Machine learning 27(3), 313–331 (1997)
11. Pazzani, M., Billsus, D.: Content-based recommendation systems. The adaptive web pp. 325–341 (2007)
12. Tax, D.: One-class classification. PhD thesis, Delft University of Technology (June 2001), `http://www.ph.tn.tudelft.nl/davidt/thesis.pdf`