Modeling ontological relationships for personalization of restaurant recommendations

Ahmed Attia, David Eng, Leonid Keselman, and Stephanie Tang

{aattia | dkeng | leonidk | svtang}@stanford.edu

Team Symmetra | March 17, 2017

Abstract

Our problem task was to build an ontological reasoning system to recommend restaurants to a user, given their preferences, ranging from dietary restrictions, budget, location, and favored cuisines. We used the Yelp Dataset to test and model realistic user preference behavior.

We approach this problem first by modeling restaurants, reviews, and user preferences. Then, we evaluate the effectiveness of three problem-solving methods: OWL modeling of our desired ontology of restaurant preferences, SQL querying of the existing data, and Bayesian networks. We build a web interface to enable efficient interaction with parts of our systems.

From our evaluations, we demonstrate the value of combining ontological modeling with unsupervised algorithms to provide rich recommendations to users, modeling the semantics and relationships between cuisines, businesses and other properties that are of interest to users.

Background

With the plethora of restaurants, it is incredibly difficult to sift through them all to find a suitable restaurant for one's tastes. Even more difficult is locating restaurants nearby when one does not even know what exists around them. Therefore, having a restaurant ontology that models aspects of a restaurant along with a model of a user preference is needed to effectively identify restaurants that best match a user's preference. To filter restaurants by a collection of restrictions is perhaps the most direct solution to finding potential restaurants; however, in addition to this method, another more effective method would be to look at what restaurants a user has previously visited and liked to identify restaurants similar to those. This allows the user to find interesting restaurants without needing to explicitly identifying what their food preference is; rather, the problem solving method can model features of the restaurant and find patterns in features that a user does like.

Existing companies like Yelp already attempt to solve this problem by allowing a user to filter restaurants by budget, cuisine, and location. However, Yelp mainly provides a platform on which users may share their reviews of restaurants, and much of the power of identifying "matching" restaurants rests in the user who must read many reviews for restaurants. A system that recommends restaurants based on non-explicit user preferences allows us to shift the burden from users to machines.

Yelp Dataset & Related Work. For the project we are using the Yelp Academic Dataset (1). It includes 800k restaurants, with 2.5M reviews. Additionally, there are 500 category tags (Chinese, Fast Food, etc.) that are applied to each business. The dataset only encompasses a dozen cities, none of which were in California. In addition to information about each business, there are reviews (with text, scores, and associated user and business identifiers), (anonymized) user profiles and information about when users "check-in" to different businesses. This dataset has been the basis of existing work, including applications of sub-topic modeling such as LDA (2) to understanding what hidden subtopics users are implicitly discussing (3). Additionally, as Yelp is such a large-scale dataset of user-interaction, it has been the basis of building systems in the clinical domain as well. For example, the corpus of patient experience was built as a way of analyzing patient experiences with clinicians based on methods first implemented, tested, and verified on the Yelp Dataset (4). This same dataset has been used to investigate large-scale analysis of health-care providers (5).

However, for our purposes, we will limit our focus to the restaurant sub-domain, as it is a domain which is more accessible to non-experts, although the tools developed here could be applied to more specific domains.

Methods

We presented multiple methods for modeling preference differences between users. Ideally we'd build methods that (1) use real, large-scale data, (2) demonstrate the value of ontological system building and (3) allow users to find restaurants more efficiently. Thus, we built multiple systems to help provide ontological structures to this data, enable efficient queries across it, and use it to build (and evaluate) recommendation systems.

Data Cleaning. In general, the Yelp dataset is very large and required cleaning and filtering to make it relevant for the scope of this project. First, businesses which lacked a *Restaurant* tag were removed, as were their respective review, check-in and users information. Second, the categorical labels in the dataset were accumulated, filtered for sufficiently high presence. Then turned from a per-business list feature to a categorical one-hot encoding for all businesses. This allowed for efficient querying of properties. For some of the classification tasks, we limited the number of categories. This was to eliminate descriptors which would not be useful in the prediction task due to their low frequency of appearance. These descriptors along with other properties (city, rating) can map directly to a series of categorical features for the representation of a restaurant which serves as input to different problem solving methods.

Ontology. Since we use data from Yelp, our ontology is dependent on the data available. From the data we have, we model the recommendation system as many different types of Restaurants. Restaurant types include different cuisines, venues, special interest restaurants, price range, and food types (e.g. a waffle shop). A Restaurant has a Name, Address, Rating, Price, and is part of one or more Categories. Restaurants also have many Reviews. A Review is written by some User and has

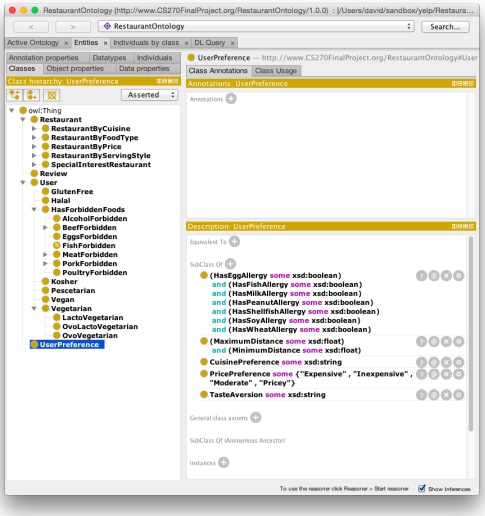


Fig. 1. A UserPreference concept as represented in OWL and visualized in Protege.

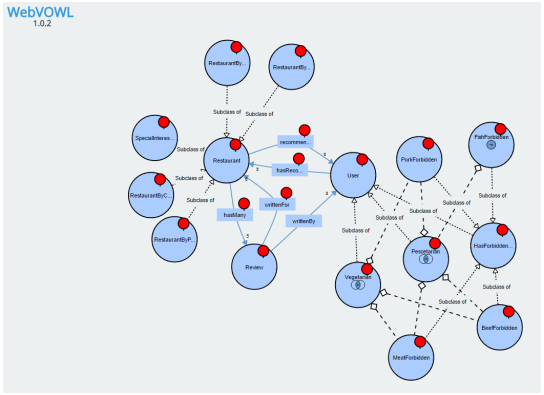


Fig. 2. A visualization of the high-level relationships between concepts.

a Description and Timestamp from when it was written. Users also have a Name, Age, and has a UserPreference. A UserPreference contains information about a User's DietaryRestrictions, including Allergies, ForbiddenFoods, and other preferential dietary restrictions like Vegetarian, Vegan, Halal, or Kosher. From a User's UserPreference, we can reason about a User's recommended Restaurants. If time permits, we may expand on the attributes of Restaurants and UserPreference to allow for more detailed and intelligent reasoning. Below is a visualization of some of the top-level concepts in our ontology. Please see our OWL file for more specific subclasses, which can be accessed via Protege. A full online visualization of the ontology can be found at http://owlgred.lumii.lv/online_visualization/295g, but due to the number of subclasses, may be difficult to read.

Problem-Solving Methods

In order to derive user recommendations, we built multiple different problem solving methods. The first is a SQL-based query system for searching through our data based on properties of interest. This mirrors the OWL ontology specified in the previous section. A second system is an recommendation system based on star classifiers. The third is a Bayesian Network to model relationships between various category types. The second is a

SQL Querying via Web App. This problem-solving method allows a user to easily search and filter through thousands of restaurants based on any of a Restaurant's attributes, including name, category, minimum price range, maximum price range, city, within a radius around a given longitude/latitude, minimum star rating. This method takes a user's input (all of which are set value types and does not support natural language queries) to dynamically generate a valid SQL query and executes the query on the existing Restaurant database. The web application then displays a list of viable restaurants

and a user can click on each restaurant to see all details/attributes of the Restaurant instance (and any data relevant to the Restaurant in any other table like Categories and Reviews). Additionally, if a location (latitude and longitude) is given with a radius, results are ordered and ranked by nearest restaurants first. Users are also able to order results by descending price and rating. The application also includes a few sample predefined users and their preferences that automatically fill in the user input boxes.

Multi-Class Classification. We built a system to understand how powerful different features are to modeling user preference. For each user's list of reviews, we held out one review and tested whether we could predict the number of stars the user gave to this review, given the remaining reviews. We filtered for users with at least 50 reviews. Random guessing missed by an average of 1.14 stars.

Using Business Categories. Using the provided business category information (such as if the business is a *Bar* or *Mexican* restaurant). A k-nearest neighbors classifier on this dataset missed by an average of 1.06 stars.

Using User Review Characteristics. We also experimented with extracting adjectives from reviews and using this adjectives to model each business. The top one thousand adjectives were used to construct vectors representing each restaurant, where a restaurant's value for position `X` in the vector is the number of times users described the restaurant with the adjective corresponding to position `X`. While we considered using LDA topic modeling methods to model this problem with lower dimensionality (2), we didn't feel our dataset would benefit from using a lower-dimensional embedding. Using these features, our classifier missed by an average of 0.982 stars with the k-nearest neighbors.

Bayesian networks. We wish to model the interactions between user preferences with a Probabilistic Graphical Model, in this case, a directed Bayesian inference network. For example, we'd like to if users who like "Thai" food also like "Chinese" food, or how one's preference for "Cafe" impacts one's view of "Ice Cream".

We built a Bayesian Network to model the relationship between different restaurant categories. Each restaurant is assigned a subset of these categories to describe the business, and they vary from cuisines (Thai, Chinese) to dietary restrictions (Vegan) to venue types (Wine Bar). We'd previously analyzed how these categories interact in our OWL ontology described above. We then built a method to select a high-quality 50 categories (described below), whose implicit relationships we modeled using a Bayesian network. We used an automated technique of building the Bayesian Network (described below), and then evaluate its performance. Additionally, we added our own large-scale ontological categories (Such as Asian Food, Nightlife, Morning, and Lunch) that combined results from other categories, and evaluated how this affected the quality of our Bayesian estimation.

Category Selection. Our initial selection of the top 50 categories to model this problem posed an issue: no users had seen all the categories, despite having hundreds of reviews of dozens of businesses (each which have multiple tags). This is because there are disjoint sets of category types. For

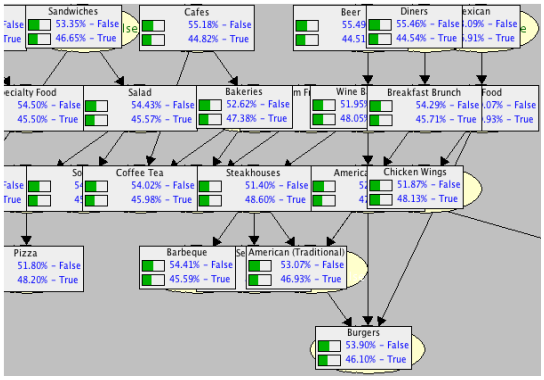


Fig. 3. Before Evidence

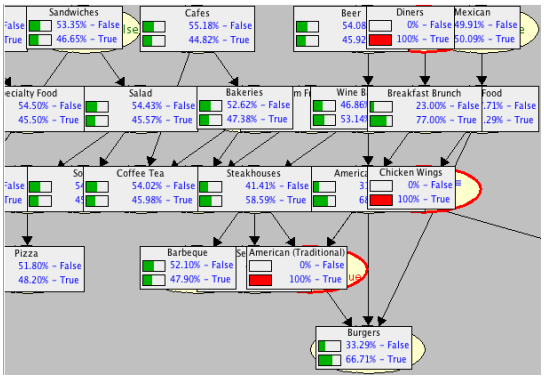


Fig. 4. With Selective Evidence

Fig. 5. An example of using our ontological model for restaurant categories via SAMIAM. With **True** selected for Diners, Chicken Wings, and Traditional American Food, we can see that Burgers increases in probability.

example, one of the most popular categories was Event Planning and catering. However, users who reviewed caterers had actually few reviews of classical restaurants. So we built a greedy expectation-maximization algorithm, where we select the category with the most users, conditioned on the subset of users who have already reviewed our previous categories. This provides us a high quality subset of categories with good user diversity and plentiful data. We then choose the top 50 categories from this list, which we find has sufficient diversity in restaurant types.

Bayesian Network Generation. Our fundamental metric for what constitutes a user liking a category is if their five-star rating for a restaurant exceeds the average score of a business. This tends to split the data to have a roughly 50%/50% prior for users liking each category (though not exactly because we used mean score not median score). Users are then filtered for those with at least 20 reviews (to speed up processing), and we're left with roughly 15,000 users, of whom only 50 have reviewed all 50 categories. Each user has a True/False/Not-available for whether their average star review for each category exceeds the average star review for that category across all users. A graph is then built by comparing the significance of these binary metrics with a chi-squared test. This is known in probabilistic graphical model literature as the PC algorithm (6). This process takes about 8 hours to run using the implementation described in (7). Once the edges are built, we simply aggregate statistics to compute conditional probability tables. Example tables are seen in figure 2 on page 12.

Interface for Bayesian Network. Once we built the models, we can export a graph and it's associated conditional probability tables in a standard XML format. These can then be loaded into tools like SAMIAM (8) (9) and Weka (10). We use Weka to perform automated layout of the graph, and then use either SAMIAM or pgmpy to do inference (11). The SAMIAM implementation is preferred for manual usage (see figure 3 and 4), while the pgmpy model is utilized in our web interface, described below. This portable XML format allows us to use the same model in multiple tools. As described in the results section, we also built a manual Bayesian Network for evaluation.

Modeling Additional Semantic Properties. Additionally, we wanted users to be able to provide other high level information about businesses, and for us to be able to provide coherent recommendations. For example, say a user gave us information about a business that they liked, or a time-of-day they liked to eat, we wanted to be able to model, rank and compare businesses using this type of semantic information. That is, we wanted to establish an ontology (or at least coherent distance metric) to understand behavior such as "People go out on Fri & Sat nights", "People each brunch Sat & Sun mornings".

For this goal, user checkin data was used to model, in an unsupervised way, the ontological similarity between attributes, such as business relationships or times of the week. In this case, we use model businesses looking at them in the high dimensional space. For example, we can use checkin counts at a time of day as our embedding space, and then perform a lower dimensional embedding (in this case, using a nonlinear method, tSNE (12), although others, like a linear PCA, would have also been appropriate). This gives us the business relationships seen in 7. It provides a sensible distance metric, for example, we can see that brunch places cluster together in the top left. This can be used to understand a semantic distance between businesses. With further analysis, these embeddings could be organized and grouped into ontologies.

By transposing the matrix mentioned above and performing the same technique, we can obtain a lower dimensional embedding relating times of the week together based on business checkins. This is shown in figure 6, and shows natural groups. For example, the top left corner shows early mornings, while the top right cluster contains "going out" times (Thur/Fri/Sat nights). This tells us, for example, that Sunday at 1am is closer (semantically) to Thursday at 8pm than it is to Sunday at 6am. This could be useful in providing restaurant recommendations by time-of-day.

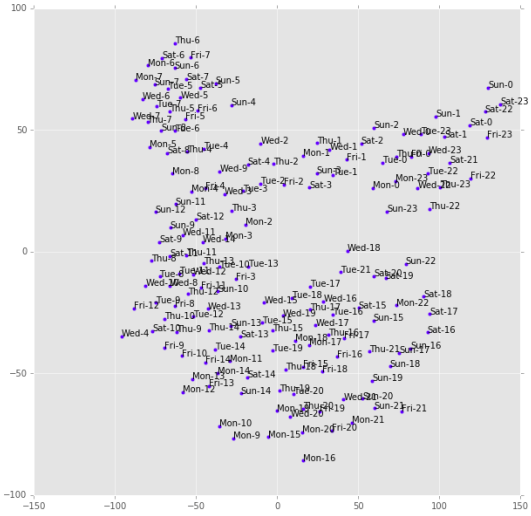


Fig. 6. tSNE embedding showing semantic distances between various times of the week.

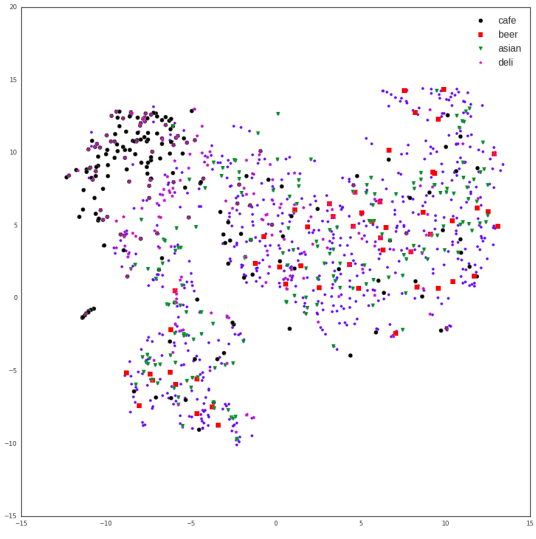


Fig. 7. tSNE embedding showing semantic distances between restaurants in Pittsburgh. We added categorical labels to show clustering.

Fig. 8. An example of modeling semantic distances between non-categorical labels.

Evaluation

SQL Querying via Web App. For the web application, a set of users/evaluators will be asked to execute any query and track whether the results they would actually consider visiting. Additionally, users will be asked to give qualitative feedback on what they like about the application and what is difficult to use based on a set of qualitative evaluation questions. Common responses (i.e. more than a quarter of users bring up) will be collected for analysis and discussion.

Multi-Class Classification. Our evaluation for the multi-class classification PSM consists of predicting the rating for each user review given their other reviews. We only use reviews coming from users who have reviewed more than 50 businesses so that their is enough data for the described PSMs. This is around 500k reviews from 5k users in total. The metric we are using to evaluate the prediction is the average difference between the true star rating in a review and the predicted star rating

Bayesian networks. We built three different Bayesian networks in order to evaluate the value of explicit ontological modeling. The first was fully automated and generated using the algorithm described above. The second had the addition of our own large-scale ontological categories (Such as Asian Food, Nightlife, Morning, and Lunch), and was again fit with the same algorithm. The last was a fully manual network drawn by us, based on the types of relationships we would expect.

In order to evaluate our models, we took a subset of users from the Yelp dataset and tried to predict their preferences with regards to various categories. We took users who had reviewed at least twenty restaurants and reviewed fairly popular restaurants (over 100 reviews) to narrow our dataset to around 1,000 users. We then would randomly drop 20% of the users' known preferences about food categories, and evaluate how well our Bayesian Networks would predict those 80% of preferences, conditioned on the evidence of how they feel about the other 80% of restaurants. This provided a quantitative evaluation of our recommendation system. Additionally, we integrated these Bayesian networks into our Web Application and SAMIAM, to allow user feedback of the qualitative variety.

Results

SQL Querying via Web App. Due to the sheer amount of reviews, we have limited the reviews to 30,000 entries, and therefore most restaurants in the sample web application do not have any associated reviews. An example restaurant with reviews has RestaurantId = DgZ-pZUo3drzpiCDlDr9IQ.The restaurants are limited to several cities, based on the original Yelp data set. With more storage space and more processing power beyond what is available on the shared computer clusters, we could expand to processing more data. Currently, the database schema only represents the Restaurant and Review concepts and associated attributes from the ontology. The search form UI captures aspects of a UserPreference concept.

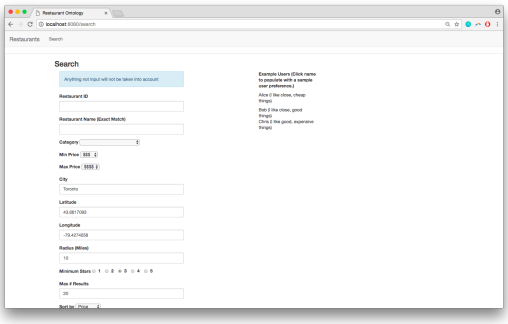


Fig. 9. Initial search form

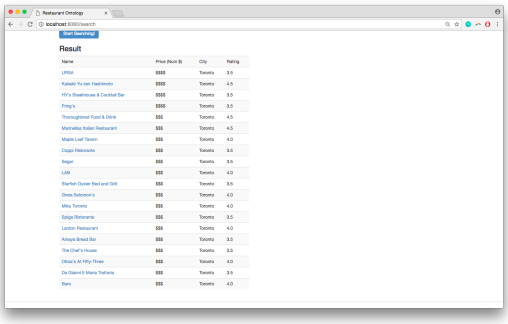


Fig. 10. Ranked results

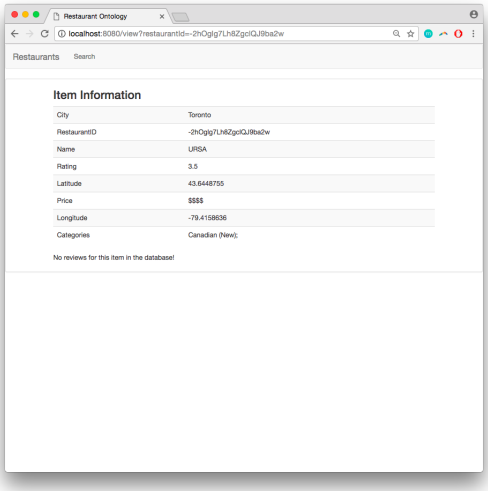
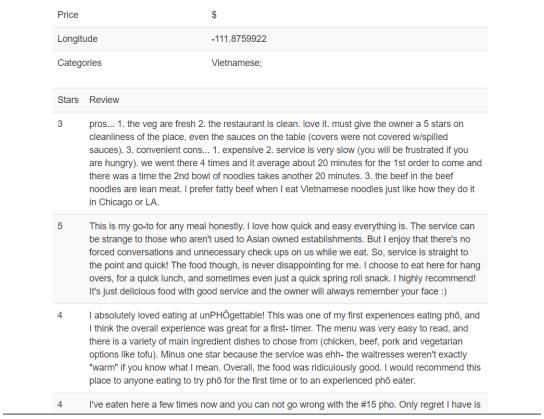


Fig. 11. A restaurant's information



Price	$
Longitude	-111.8759922
Categories	Vietnamese;

Stars	Review
3	pros... 1. the veg are fresh 2. the restaurant is clean. love it. must give the owner a 5 stars on cleanliness of the place, even the sauces on the table (covers were not covered w/spilled sauces). 3. convenient cons... 1. expensive 2. service is very slow (you will be frustrated if you are hungry). we went there 4 times and it average about 20 minutes for the 1st order to come and there was a time the 2nd bowl of noodles takes another 20 minutes. 3. the beef in the beef noodles are lean meat. I prefer fatty beef when I eat Vietnamese noodles just like how they do it in Chicago or LA.
5	This is my go-to for any meal honestly. I love how quick and easy everything is. The service can be strange to those who aren't used to Asian owned establishments. But I enjoy that there's no forced conversations and unnecessary check ups on us while we eat. So, service is straight to the point and quick! The food though, is never disappointing for me. I choose to eat here for hang overs, for a quick lunch, and sometimes even just a quick spring roll snack. I highly recommend! It's just delicious food with good service and the owner will always remember your face :)
4	I absolutely loved eating at unPHÖgettable! This was one of my first experiences eating phô, and I think the overall experience was great for a first- timer. The menu was very easy to read, and there is a variety of main ingredient dishes to chose from (chicken, beef, pork and vegetarian options like tofu). Minus one star because the service was ehh- the waitresses weren't exactly "warm" if you know what I mean. Overall, the food was ridiculously good. I would recommend this place to anyone eating to try phô for the first time or to an experienced phô eater.
4	I've eaten here a few times now and you can not go wrong with the #15 pho. Only regret I have is

Fig. 12. An example review

Fig. 13. An example of the different web page views available within our web application. This allows users to specify and view the query results in a friendly and accessible way

To evaluate the SQL-based web app, we asked 10 people to try out some queries. We then asked them to give qualitative feedback on the system, answering the following questions

1. Did this application answer your information need?
2. Was the application easy and intuitive to use?
3. Were the results returned in a useful format?
4. Would you visit the restaurants returned?
5. What did you want to do with this app that you could not?
6. What features were frustrating or unintuitive to use?

Common answers and our analysis of the response:

- *Ranking the restaurants by distance took a really long time (more than a couple minutes)*
  Analysis:The version of the query this user used attempted to order several thousand restaurants by distance, which is not optimized. We altered this query to be a nested query such that the other filters like city, name, ranking etc were applied first on the inner query, and then a second outer query would order the smaller result set.

- *UI requires user to fill out preferences every time - might be nice to save it?*
  Analysis: This comment suggests future work on the web app to further expand the ontology to additionally save user preference information. This could be done by adding a Users table with columns that store preferences. Our current application only has some sample saved user preferences.

- *Names were case-sensitive and I couldn't search by keywords.*
  Analysis: Exact search on strings is maybe too limiting. While this is faster, this query made it difficult for the user to find restaurants by name or city if the capitalization varied. In a future iteration of the web app, we could consider using partial string matching.

- *The results returned are 80-85% of the time potential restaurants users would go to and does answer their question, but this is due to the web app simply meeting the most basic conditional requirements*
  Analysis: The PSM does answer the user question/information need, but not as well as they would want it to. They want more expressivity in their queries. It seems like users would like even more ways to filter the data and perhaps in a "fuzzier" way. The Yelp data also includes adjectives that describe restaurants. If we incorporate this into the database, users could potentially to search by some keyword like "home-style" rather than just the restaurant name or predefined category type.

- *Being able to see more reviews would be nice*
  Analysis: This is a data issue, and less so about the PSM itself. With more storage and processing power, we could expand this database to include many more restaurants from different areas and their associated reviews.

- *Why are there no addresses?*
  Analysis: Understandably, no one actually knows their latitude and longitude. While this is easy to store in a database, maybe we can use a geo plugin (for data-interoperability) that would allow us to convert addresses into the latitude/longitude our queries need while allowing users to express location in a more human-friendly manner.

Other Observations from User Testing:

- We had most of the users try to search Palo Alto (which the Yelp data we used didn't actually cover) so obviously expanding the data set would be helpful to the user.

- Some more specific user preferences like allergies were not taken into account in this application. If we expanded the schema such that each restaurant to have a menu with dishes consisting of ingredients, we could do more complex queries, though this may be slow.

Multi-Class Classification. Our baseline for the prediction of user ratings was random guessing over the distribution of a user's reviews. That is, the rating predicted for an unseen review was randomly chosen from a user's reviews. This method missed by an average of 1.14 stars. The next method we used was nearest neighbor using business categories (like 'Asian Food', 'Diner'). This approach performed moderately better, missing by an average of 1.06 stars. The final method we tried was a nearest neighbor approach with business adjectives, which missed by an average of .982 stars. The conclusion we draw from this is that it is more useful to look at the properties of a business as assigned by other users (the adjectives were scraped from a business's reviews) than the business categories which are given by yelp.

Bayesian networks. As explained in the previous methods and evaluation sections, we built three different Bayesian Networks to model user preferences between categorical labels such as "Chinese", "Thai", "Mexican". We then used a subset of user data to see how well our models are able to predict user preferences, given some of a user information. We tested three different networks for the same nodes, a fully automatic one built on chi-squared independence tests, a fully manual one we created ourselves, and a semi-supervised one, that includes some manual heuristic categories (such as Morning, Night, Lunch) but then follows the same automated network building as the first model. Quantitative results are shown in table 1, while a qualitative understanding can be derived from analyzing the conditional probability tables, such as the example ones shown in table 2.

In general, we show that the automated algorithm constructs a better Bayesian Network than one drawn by a human. However, if a human adds higher level meta-categories to the data, it effectively adds additional ontological information. This expanded Bayesian network can use extra nodes to better fit the data. This expanded network performs even better. We therefore have demonstrated that injecting hierarchical or ontological data can improve the quality of your models. In the future, we'd like to explore clustering methods in order to generate these additional hierarchical node structures.

Table 1. Quantitative Results, comparing our automatic, manual, and semi-supervised networks

Inference Model	Accuracy
Random Guessing	50.7%
Majority Guessing	63.5%
Bayesian Network (Manual)	65.4%
Bayesian Network (Automatic)	67.8%
Bayesian Network (Semi-supervised)	72.4%

Random is the result of guessing. Majority guessing is the the result of always guessing False. Manual is a MLE-fit to a Bayesian Network drawn by a human. Automatic uses the PC Algorithm to construct the graph (7). Semi-supervised adds human-constructed meta-categories to the dataset, and then uses the PC algorithm.

Discussion/Future Work

Based on user feedback for the SQL-based web application, future work could further expand the database schema to save user and user preference data. Other useful features that would help the system better meet users information need include: 1) using actual addresses instead of just longitude/latitude, 2) using similarity string matching instead of exact match and further optimizing the resulting query, 3) normalizing data to be case-insensitive and removing special characters, 4) expanding the restaurant data available in the database, and 5) expanding restaurant ontology and database to include menus, dishes, and ingredients to better find restaurants by food restrictions.

Moreover, the OWL ontology we created in Protege includes various categorical information we would like to see, but isn't currently present in the Yelp Data. For example, information about higher level ontological concepts (like "Morning" foods), or dietary restrictions (like Kosher or nut allergies). If we could acquire menu data, it could allow for more stringent filtering of foods & dietary restrictions. Additionally, we'd like to return rankings with a more options for different semantic distances than those outlined here. For

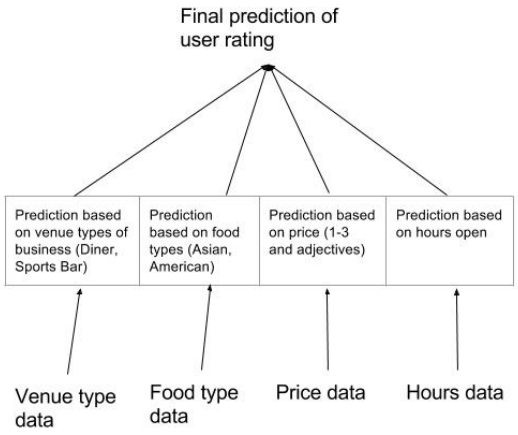


Fig. 14. An example of our targeted integration of user information into a successful query.

example, one keyword type could be 'venue' (like sports bar or diner, large/small, etc) and another would be 'type of food' (like Asian, Greek). For example, see figure 14.

The goal behind this type of modeling is to bridge the gap between the type of data that is easy to enter, track, and maintain, with the types of information users know and would like to provide search systems. That is, by having a good ontology, and the ability to reason semantically, we can provider users richer recommendations. This would be possible even when we have only a very incomplete image of their preferences. We have demonstrated such systems by building semantic distances between times of the week, businesses themselves, and a robust Bayesian Network between different food categories. There is much need and interest to expand on this work, as these types of systems are able to provide users insight into their own desires and needs in seemingly percipient ways.

Division of Labor

- Ahmed - Multi-class classification with user review data.
- David - Multi-class classification with business categories, expanding queries + sample users.
- Leo - Data cleanup, EM algorithm for category selection, all Bayesian Network methods, including modeling, fitting, manual ontology for Bayes Net, evaluation and integration into the web application. Semantic embeddings . Writing of final report.
- Stephanie - Restaurant ontology + database schema, data parsing for SQL database, web application access point + server/generating queries with user input. Writing of final report. General organization, scheduling, planning.

Table 2. Example Conditional Probability Tables with real probabilistic relationships from the Yelp Dataset. True means the user likes it more than the average user. False means the user likes it less than the average user.

P(Beer|Wine)	Wine(False)	Wine(True)
Beer(False)	0.48	0.20
Beer(True)	0.52	0.80

P(Chicken Wings|Fast Food)	Fast Food(False)	Fast Food(True)
Chicken Wings(False)	0.58	0.31
Chicken Wings(True)	0.42	0.69

P(Coffee & Tea | Bakeries, Cafes)	Bakeries(False) Cafes(False)	Bakeries(False) Cafes(True)	Bakeries(True) Cafes(False)	Bakeries(True) Cafes(True)
Coffee & Tea(False)	0.60	0.24	0.38	0.08
Coffee & Tea(True)	0.40	0.76	0.62	0.92

Appendix/Program Submission

Github Link. All of our code is available at https://github.com/gnedivad/yelp

Bayesian network. To evaluate the Bayesian Inference models, simply load the `bayes453.bif.xml` (automatic), `bayesMAN3.bif.xml` (manual) or `bayes533.bif.xml` (automatic with additional manual categories) in SAMIAM (8). Additionally, one can look at the semantic similarity, model fitting algorithms, hyperparameter selection, and evaluation of different Bayesian networks `fit pgm` iPython Notebook.

Web App and SQL Interface. To run the web app server locally:

1. Ensure RestaurantsOld.db is present. This is the database file we will query from. If it does not exist or is corrupt, you will need to get the required .dat files from us (restaurants.dat, categories.dat, reviews.dat, not included in the submission due to memory size) and recreate the database as such:
   (a) `sqlite3 RestaurantsOld.db < create.sql`
   (b) `sqlite3 RestaurantsOld.db < load.txt`
   (c) `RestaurantsOld.db` should be in the same folder as `sqlitedb.py` and `testServer.py`
   (d) Double check that line 3 of `sqlitedb.py` reads as:
   `db = web.database(dbn='sqlite', db='RestaurantsOld.db')`
   The names of the database file must match the second parameter!
2. Ensure all dependencies are installed, including pandas, pgmpy, pyparsing, json, sys, and os. The lib folder already includes web.py, sqlite3, MarkupSafe, and jinja2
3. Start the local server by running `python testServer.py`
   (a) You can now visit the valid links in your local web browser! Most likely, the base url will be http://localhost:0.0.0.0:8080/ but the port number may be different.
   (b) Try out the SQL querying at http://localhost:0.0.0.0:8080/search
   (c) See Bayesian networks at http://localhost:0.0.0.0:8080/bayes

To visit the web app live, visit http://web.stanford.edu/~svtang/cgi-bin/cs270/testServer.py/search
Note that only the search feature works live. Cgi-bin does not cooperate with the necessary dependencies for the Bayesian probability feature. The Bayesian probability feature can only be accessed by locally running the server.

References

1. Yelp (2013) Yelp's Academic Dataset (https://www.yelp.com/academic_dataset).
2. Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.
3. Huang J, Rogers S, Joo E (2014) Improving restaurants by extracting subtopics from yelp reviews. *iConference 2014 (Social Media Expo)*.
4. Rastegar-Mojarad M, Ye Z, Wall D, Murali N, Lin S (2015) Collecting and analyzing patient experiences of health care from social media. *JMIR Res Protoc* 4(3):e78.
5. Felix C, Pandey AV, Bertini E, Ornstein C, Klein S (2015) Revex: Visual investigative journalism with a million healthcare reviews in *Proceedings of Computation+ Journalism Symposium (CJ)*.
6. Spirtes P, Glymour CN, Scheines R (2000) *Causation, prediction, and search*.
7. Koller D, Friedman N (2009) *Probabilistic graphical models: principles and techniques*. (MIT press).
8. Darwiche A, et al. (2004) Samiam: Sensitivity analysis, modeling, inference, and more. *Software available from http://reasoning. cs. ucla. edu/samiam*.
9. Darwiche A (2009) *Modeling and reasoning with Bayesian networks*. (Cambridge University Press).
10. Hall M, et al. (2009) The weka data mining software: an update. *ACM SIGKDD explorations newsletter* 11(1):10–18.
11. Ankur Ankan, Abinash Panda (2015) pgmpy: Probabilistic Graphical Models using Python in *Proceedings of the 14th Python in Science Conference*, eds. Kathryn Huff, James Bergstra. pp. 6 – 11.
12. Maaten Lvd, Hinton G (2008) Visualizing data using t-sne. *Journal of Machine Learning Research* 9(Nov):2579–2605.