Mining User Reviews to Recommend Popular Dishes

Sonia Mohanlal
University of Illinois
Urbana-Champaign
Computer Science
mohanla2@illinois.edu

Namrata Prabhu
University of Illinois
Urbana-Champaign
Computer Science
nprabhu2@illinois.edu

Xiaocheng Yuan
University of Illinois
Urbana-Champaign
Computer Science
xyuan12@illinois.edu

ABSTRACT

With the rapid growth of the internet as well as increasingly more accessible mobile devices, the amount of information being generated each day is enormous. We have many popular websites such as Yelp, TripAdvisor, Grubhub etc. that offer user ratings and reviews for different restaurants in the world. In most cases, though, the user is just interested in a small subset of the available information, enough to get a general overview of the restaurant and its popular dishes. In this paper, we present a way to mine user reviews to suggest popular dishes for each restaurant. Specifically, we propose a method that extracts and categorize dishes from Yelp restaurant reviews, and then ranks them to recommend the most popular dishes.

1. INTRODUCTION

Gone are the days when information was hard to find. With the advent of the internet, information has become ubiquitous and instant. Now, with the high volume of information, users face the problem of information overload. User review data for restaurants is one of the many large data sources available online, and contained inside the review data is valuable information about the user’s interests, which reflects a great deal about the underlying entities. However, most of this data is unstructured. Thus, the vast proliferation of reviews necessitates a filtering mechanism to extract information that meets users information need.

Currently, thousands of restaurant reviews are being generated on websites like Yelp, TripAdvisor, Grubhub etc. around the world each day. Many of these applications host review information that provide valuable information to customers. One such valuable information would be providing popular dishes from restaurants for each user, especially for restaurants with a large amount of reviews where users face the problem of information overload. Providing a ranked list of popular dishes would help non-local traveler get a quick overview of the restaurant and recommended dishes. Such information is also helpful to restaurants as they would get an idea about the popular dishes among users. However, current tools do not meet this information need adequately. In many existing ranking algorithms, it is required to have a network built on top of the entities, with edges representing connections between entities and their importance. A ranked list of important entities of a certain type could then be extracted from the network using the connections and weighting between entities. This however, is not available in the case of food entities. The primary data source available is a large number of unstructured data in restaurant reviews, each consisting of a piece of review text and a rating. Although it is possible for us also construct a network on top of the available data, it is better to directly mine information from the corpus for some of the important information could be omitted during network construction, such as user sentiments towards entities.

In this paper, we propose a novel popular dish extraction framework that operates on top of a large text corpus of user reviews for restaurants. We first extract entity mentions from the text corpus. Then we perform a set of techniques to effectively remove unwanted types form the extraction. Finally we combine frequency, explicit user ratings, implicit user sentiment, as well as recency attributes of entities to perform ranking on the extracted food items, to obtain a ranked popular dish list for restaurants.

To take advantage of the available large text corpus, we leverage information that is available in the entire set of documents to help us find quality food entities, by applying techniques including POS tagging, sentiment analysis, and document frequency. We construct a ranking function for each item extracted that “gathers” inputs from reviews associated with each dish, employing a positive/negative feedback scheme, with an exponential decay to model recency of reviews and their impact.

The contributions of this paper are summarized as follows:

- We perform an analysis on current techniques and tools...
that we design methods to effectively remove noisy and unrelated phrases from extracted food entities.

- We take advantage of the underlying data source and rank dishes using both explicit and implicit information derived from review text corpus.

A system[10] was previously described to perform automatic extraction of multiword units from user reviews for businesses. The system employs a semi-supervised approach, by using a list of seed phrases which indicate the category of interest. The method proposed first generates candidate multiword aspects by performing POS tagging on the review, and then merging adjacent words by looking at the normalized pointwise mutual information. A second round of merging is then performed by checking association between merged candidates, examining the relations. The seeded words are used to compute a list of features by generating dependency triples of appearances which captures the grammatical context of the seeded words in the corpus. Next, the feature set is used to compute similarity scores for all the candidate multiword aspects, which will then produce a ranked list of candidates for each seeded word/phrase. Finally, the authors calculate the likelihood of each extracted multiword aspect belong to which category by weighing the MWU scores based on the NP-Score which is calculated using the POS structure. This work produces some desirable results in extracting food items from large corpus of text reviews, by using similarity measures. While the results capture contextual information between candidates and the seeded words, the actual type is not always perfectly reflected by the underlying context. Moreover, the focus of the paper is on extracting entities of any seeded types. Our goal in this paper is to propose a method that extracts high quality dishes and then rank the popularity of those dishes based on information reflected from the underlying reviews.

2.2 Related Work

During our research we discovered that, currently there are no existing applications which specifically ranks and lists the popular dishes in a restaurant. For example, when we look up a specific restaurant on Yelp, the highlighted keywords which should mention something unique about the restaurant, are unhelpful and quite unrelated. This would definitely pose a problem for users who are specifically trying to decide what to order during their first dining experience. Moreover, to get a decent idea of what the popular dish might be, the user would have to crawl through several reviews about the restaurant, which could be tedious and ultimately fruitless.

Another example which we came across during our research, was DishTip which, similar to our problem definition ranked the most popular dishes in a user queried restaurant. However this information proved to be incorrect and at times unavailable. For example, while querying for Sakanaya, a popular Japanese restaurant in Champaign, DishTip returned no results and didn’t even recognize Sakanaya as a restaurant. Also, while querying the popular brewery Destihl, their most famous dish (Beer battered asparagus) which had over 37 user reviews was ranked 4th on the list, with Ice Tea with only 6 user reviews was ranked 2nd. These results also made no mention of Destihl’s alcoholic beverages which is unusual for a well known brewery. These examples clearly indicate, that there aren’t many application which specifically address this issue. With the large amount of user reviews regarding a restaurant, it should be relatively easy to find popular dishes making it possible for users to have a pleasant dining experience without scrolling through several hundreds of reviews.

2.3 Problem Definition

The input to our paper is a corpus of user reviews for restaurants in a particular region. From that corpus, we aim to utilize the following:

- Dish mentions in reviews
- Review Rating
- Sentiment of dish in review
- Timestamp of review

Our output would be a ranked list of recommended dishes.
3. DISH ENTITY EXTRACTION FROM REVIEWS

In this section, we explored several methods to extract entity mentions from reviews. We subdivided our dish extraction to two different parts, the step would involve phrase mining to obtain common entities or common keywords from review data. The second step would involve categorizing/filtering entities related to menu items using natural language processing techniques.

3.1 Phrase Mining from Review Text

In this section, we explored several different techniques to extract relevant entities and phrases from each restaurant. Some of the techniques we explored include collocation mining with pointwise mutual information, SegPhrase[7] and TopMine[6]. As we ran our experiments, we found that results from collocation mining performed significantly worse than SegPhrase and TopMine giving us both lower precision and recall. In this section, we describe the two major algorithms we used to mine phrases.

3.1.1 TopMine

We first exploited TopMine[6] to extract entities given a corpus of user reviews for a restaurant. According to our data model, each review for a restaurant represents a document. Thus TopMine first segments each document into unigrams and n-grams. It then applies Latent Dirichlet Allocation (LDA) topic model on the segmented document. Finally the algorithm estimates the topic model hyperparameters by using Gibbs sampling. The final output of this algorithm would include the distribution by topic of single and multi-word phrases. The input to TopMine would be the extracted user reviews for each restaurant which has been mentioned above. Thus we run the algorithm for the set of reviews from each restaurant using the following configuration:
• minimum phrase frequency: 4
• maximum size of phrase (number of words): 6
• number of topics: 1
• Giggs sampling iterations: 1000
• significance threshold for merging unigrams into phrases: 2
• burnin before hyperparameter optimization: 100
• alpha hyperparameter: 2
• optimize hyperparameters every n iterations: 50

After running the algorithm, we inspected the resulting topics. The top phrases did include popular dishes from each restaurant (for example the results from Black Dog included "burnt ends" which is its most popular dish), as well as other common phrases such as "food was good", "Champaign", "Chambana". Thus our results did include a significant amount of noise which needed to be filtered out.

3.1.2 SegPhrase
Given a corpus of user reviews, SegPhrase[7] first segments the entire corpus and selects those phrases which are higher than the given threshold support. The algorithm also requires as input, a set of phrases labeled as either good quality(1) or bad quality(0). This allows the algorithm to rectify phrase counts to extract phrases which are closely related to the positively labeled phrases and ignore the ones related to the negatively labeled phrases. Similar to TopMine[6], we ran the algorithm on the extracted user reviews for each restaurant. We also constructed a set of around 300 phrases and labeled them as either good quality or bad quality. Since we’re primarily focused on extracting entities related to dishes, the good quality phrases included some example dish names while the bad quality phrases included a mixture of location names, user’s opinions and some random users’ comments.

We cloned the SegPhrase[7] repository on GitHub and applied the tool using default parameters except for using a support threshold of 3.

Overall, the results from both TopMine[6] and SegPhrase[7] were comparable. The latter gave us a higher precision with less noise, while the former gave us a higher recall with more noise. Ultimate, we picked TopMine since we determined that recall was more important than precision, especially since after extracting entity dishes, we planned to re-rank dish mentions through our ranking function.

3.2 Categorizing and Extracting Dishes from Mined Phrases
The second part of our dish extraction step is to categorize and extract only those entity mentions related to dish items. For this step, we wanted to filter out unrelated phrases such as "food is good", or "chinese restaurant" to obtain food items. This is essentially an entity typing and selection problem on the extracted list of phrases. Our first approach to solve this was to use New York public library’s food dictionary containing menu items since 1850’s [1] to filter out dish mentions. However, this gave us a low recall, because dish names are not common and similar at all across restaurants, as different restaurants name their dishes in their own way to reflect their specialties. We realized that after looking through online knowledge bases, we were unable to find a Knowledge Base that effectively recognizes food entities given our requirements. Another option would be to use learning through type propagation with the help of a distant knowledge base, like ClusType[9, 8] to filter out entity mentions related to food. Unfortunately, we found no knowledge-bases that contained typed information for food except for Freebase which got deprecated in 2015. We tried running ClusType without food as a target type as a technique to filter out entity mentions unrelated to food, however that process was time-consuming and gave us a lower recall than other techniques.

Instead of actively trying to recognize food entities from our candidates, we aimed to achieve this goal by removing entity and phrases not related to dishes. We experimented with three different techniques to remove three types of noise from our data.

3.2.1 Partial Sentences and POS Tagging
After performing the first step to extract keywords out of the text corpus, we have noted some unrelated entries in our list to be a short sentence. That is, some short phrases or sentences that were used to describe some attributes of a restaurant are used by many people and therefore extracted. Examples of such entries include “wait to go back”, or “highly recommend”. After observing the pattern of such entries, we realized that their structure are different from the dish names we are trying to extract. It is natural to think that all dish names should be nouns, so we started carefully examining some examples. For the instances we examined, we discovered that food entities all end with a noun(might contain verb in the beginning, but end with noun), while the unrelated entities mentioned above hardly ever end with a noun.

With this realization, we added a filtering step after extracting phrases from the corpus, that employs POS tagging to filter out partial sentences. We used a POS tagging tool [2] to first tag the phrases extracted in the first step, and then we inspect each tagged phrase and discard any entries whose last term is not a noun.

3.2.2 Location and Organization Entities
Besides partial sentences from the corpus that contained no food information, sometimes we also see phrases that indicate locations or organizations get extracted from the corpus. Reviews are comprehensive collections of some attributes regarding the restaurant, and therefore things like location or the organization itself would appear in the extracted phrase list for they are likely to be mentioned frequently in the review. These entities would not be filtered out by the POS tagging step because location and organizations tend to be nouns themselves.

Although we find it hard to recognize every food entity with existing Named Entity Recognizers(NER), there are many
NER tools that solve the problem of recognizing entities of type location. We experimented with a few different tools, and ended up using IBM’s Alchemy API [4] to extract entities related to location/organization.

3.2.3 Common Phrases and TF-IDF
In the last technique, we used TF-IDF to filter out keywords that were common but not relevant for our purpose. An example of such a keyword would be “Good food” or “food is good” - these keywords are fairly general and do not relate specifically to the restaurant. Our intuition for filtering such keywords was to calculate the number of mentions across the entire yelp corpus of reviews for a particular region since we realized the phrases that were extremely common is unlikely to relate specifically to the restaurants menu items.

4. RANKING DISH ENTITIES
Once we have extracted the dish names from the corpus of restaurant reviews, we now need to effectively rank these entities, in order to list the top-5 dishes for each restaurant. Intuitively, the frequency of the dish entity appearing in the reviews could provide a rough estimation of how “popular” the dish is among customers. For each dish, we collect the number of times it has been mentioned in the review for that business. However, higher frequency does not necessarily mean a higher "goodness", because reviews are not necessarily praising the dish. As a result, we do not simply look at the frequency of an entity appearing. Two other measures are important in determining the popularity of the dish namely, the overall user rating of the review as well as the user opinion of the dish within the review.

Each user, in his review, rates the restaurant on a scale of 1 to 5 stars. A user may also mention specific dishes which he liked or disliked in said restaurant. We consider both these measure to be equally important in determining the rank of the dish. For example, a user could have had a bad experience at a particular restaurant either due to poor service or long waiting times, but could have specifically enjoyed one of the restaurant’s popular dishes. It could also be possible that a user had a very good experience at a restaurant, but didn’t particularly enjoy the restaurant’s most popular dish. To provide even more accurate results on top of what we are able to find with dish frequency and user feedback, we take into account the effects of time and recency.

4.1 True Document Frequency
When extracting candidate phrases from the text corpus, we view each piece of review text as a document for a business, and all reviews for that business consists of the document collection. For each potential candidate extracted, we keep track at the generation phase in which document it is mentioned. With this information, we could easily obtain the number of mentions a dish $D_i$ has for a certain restaurant, we denote this as $C_i$. It is important to note that while calculating document frequency, we measure the true document frequency of each phrase, especially if one of the keywords is a substring of another. For example: suppose our list of keywords includes both "sweet potato" and "sweet potato fries", we calculate the true document frequency of "sweet potato" by counting only those occurrences where it is not appended by "fries". This allows us to evaluate the true frequency of the phrase. If the number of reviews mentioning a particular dish is higher, it is more likely that the dish is important. However, we also need to consider whether the
Ranking of a dish is being mentioned in a positive or negative context, since dishes which have been mentioned several times yet in a negative context would be considered unpopular.

4.2 User Rating
Given the reviews mentioning a particular dish, we can obtain the user rating of each review and assign a score based on whether the rating was high or low. For the purpose of this project, we consider ratings of 4 and above to be positive and assign it a score of 1, while user ratings of 3 and below are assigned a score of 0.

\[ \text{RatingScore} : f = \begin{cases} 0, & \text{if userRating} \leq 3 \\ 1, & \text{otherwise} \end{cases} \]

4.3 Sentiment Analysis of Review
For each mention of the dish within the review, we need to obtain the sentiment score of the dish. This information would tell us if the dish was mentioned in a positive, negative or a neutral context. Thus we extract sentences from the review, which mention this dish. We then used Stanford’s NLTK <fix this part> to obtain the sentiment of the sentence. Based on whether the overall sentiment is positive, negative or neutral we assign a sentiment score of 1.0 or -1 respectively.

\[ \text{SentimentScore} : f = \begin{cases} 1, & \text{if sentiment of dish is positive} \\ 0, & \text{if sentiment of dish is negative} \end{cases} \]

4.4 Recency
While sometimes a restaurant would have served their popular specialty dish for a long time, it is possible that some dishes existed for a while and then were removed from their menu, and if the dish was extensively mentioned in previous reviews, it could be captured by our ranking algorithm while it actually is not available or that popular any more. The recency factor in our ranking algorithm helps solve this problem. The recency of a document can be represented by time elapsed in months which is defined as:

\[ \Delta t = t_{current} - t_{document} \]

Here \( t_{document} \) is the timestamp associated with the review. Given the elapsed time, we model our exponential decay as follows:

\[ e^{-\lambda(\Delta t)}[3] \]

Here lambda is a parameter that the user can control, the higher the lambda the more dishes mentioned in older reviews are penalized and vice-versa. A lambda value of 0 would not take recency into account.

The final ranking score of a review would be a combination of the above measures. Thus the final ranking score of a dish \( d \) mentioned in a corpus of reviews \( R \) would be given by:

\[ \text{Ranking}(d_i) = \sum_{r \in R} \frac{\text{sentiment}(d_i, r) + \text{rating}(r)}{2} e^{-\lambda(\Delta d_i)} \]
Table 4: TopMine results on Destihl

<table>
<thead>
<tr>
<th>Dish Entities for Destihl after TopMine</th>
</tr>
</thead>
<tbody>
<tr>
<td>beer battered bacon</td>
</tr>
<tr>
<td>cheese curds</td>
</tr>
<tr>
<td>pretty good</td>
</tr>
<tr>
<td>good food</td>
</tr>
<tr>
<td>great food</td>
</tr>
<tr>
<td>waiting minutes</td>
</tr>
<tr>
<td>bison burger</td>
</tr>
<tr>
<td>downtown champaign</td>
</tr>
<tr>
<td>deviled eggs</td>
</tr>
<tr>
<td>beer battered asparagus</td>
</tr>
</tbody>
</table>

Table 5: Dish Entity mentions after filtering results from TopMine on Destihl

<table>
<thead>
<tr>
<th>Dish Entities for Destihl after Post Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>beer battered bacon</td>
</tr>
<tr>
<td>cheese curds</td>
</tr>
<tr>
<td>bison burger</td>
</tr>
<tr>
<td>deviled eggs</td>
</tr>
<tr>
<td>beer battered asparagus</td>
</tr>
<tr>
<td>beer selection</td>
</tr>
<tr>
<td>stout ice cream</td>
</tr>
<tr>
<td>chocolate cake</td>
</tr>
<tr>
<td>fried egg</td>
</tr>
<tr>
<td>fish and chips</td>
</tr>
</tbody>
</table>

results, there is still room for tuning to get customizable results for special circumstances or a different cultural region. For example, in a city where food taste and style changes and evolves very quickly, we might want to penalize old reviews more than we usually do. By changing the lambda value in the recency weighting equation, we would be able to change how time elapsed since the review affects the ranking contribution of that review. This way, we would be able to tune the algorithm to return results that fit our exact needs.

5.5 Results

As shown in Table 4, our algorithm was able to output the same top 1 dish for Destihl while having the 2nd and 3rd dish different. For BlackDog, our results are the same as the crowdsourced results, but in different order. Granted that our precision (if every result matches exactly in the same order) did not look very good judging from the evaluation result, it should be noted that the result we are comparing our output to is a crowdsourced list of ranked dishes. While our output was not in the final crowdsourced result, our friends agreed with us that our output is also very reasonable and some have actually provided the same list as we did (but was not displayed due to how we aggregate the crowdsourced results). Results also could have looked different had we been able to acquire a larger set of people to participate in the ranking, which should reflect the actual result better.

The above results indicate that for Destihl we obtain a precision of 0.6667 while for BlackDog we obtain a precision of 1 with a slightly different order of ranking. For both cases, we predicted the top dish accurately.

6. CONCLUSION

In this paper, we introduced a novel framework that effectively extracts food item mentions from large corpus of restaurant review text, and then ranks the food items for popularity based on the review corpus, combining user ratings for the restaurant as well as users’ sentiment towards the mention itself. Our framework is review driven and takes in input in a very common and manipulatable format. Our problem was two-fold: First we have to extract quality food items from the text corpus, and then with the extracted dishes we need an effective algorithm to rank them by popularity, eventually arriving at a list of ranked dishes for a restaurant. The key idea in extracting food items is that instead of actively looking for food entities which we couldn’t do well without the help of a comprehensive distant knowledge base or a reliable learning/classifying method, we eliminate all other kinds of unrelated candidates to obtain the list of food items. For ranking, apart from looking at the naive measure of frequency, we look at both explicit user rating and implicit user sentiment mined from the review text, finally combining the results using the effect of recency. Our work provides a sensible recommendation for popular dishes at restaurants, which have not been effectively done previously.

7. FUTURE WORK

In the future, we would like to rank dishes based on their sub-categories. For example, for a particular restaurant, we would like to find the most popular drinks, appetizers, entrees, etc. Doing so, would probably require parsing the restaurant’s menu to find the sub-category for each dish, and then subsequently extracting entities based on these sub-categories. We would also like to experiment with more
Table 6: Our results compared with crowdsourced ground truth *Our results **Crowdsourced

<table>
<thead>
<tr>
<th></th>
<th>Destihl*</th>
<th>Destihl**</th>
<th>BlackDog*</th>
<th>BlackDog**</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beer Battered Asparagus</td>
<td>Beer Battered Asparagus</td>
<td>Burnt Ends</td>
<td>Burnt Ends</td>
<td></td>
</tr>
<tr>
<td>Beef Battered Bacon</td>
<td>Cheese Curds</td>
<td>Pulled Pork</td>
<td>Sweet Potato Fries</td>
<td></td>
</tr>
<tr>
<td>Cheese Curds</td>
<td>Chorizo Stuffed Dates</td>
<td>Pulled Pork</td>
<td>Pulled Pork</td>
<td></td>
</tr>
</tbody>
</table>

techniques to improve ranking and information retrieval. One possible way to do would be to include user queries. For example, if a person is a vegetarian, we would only return the most popular vegetarian dishes for the restaurant the user is looking into, since this particular user would not find it very useful if the majority of dishes recommended to him are things he can't eat.

Also, more research could be done onto extracting high quality entities of a target type. In our methodology we still require the use of an NER tool which is web based, which if broken could affect the results that we get (the Freebase API has been deprecated and has had an impact on ClusType). The paper on extracting targeted types in reviews could be a potential direction to look into for it uses seeding to detect target types, but modifications will need to be performed to overcome the problems with contextual dependencies.

8. ACKNOWLEDGMENTS
We would like to thank the Professor Han and the course staff of CS512 without whose support this project would not have been possible.

9. REFERENCES