Picking Out Good Dishes from Yelp

Angela Gong
Dept. of Computer Science
agong@stanford.edu

Jennifer Lu
Dept. of Computer Science
jenylu@stanford.edu

ABSTRACT

Sifting through restaurant reviews on Yelp to find good dishes can be quite difficult, especially when reviewers use colloquial terms for the dishes instead of actual menu names. However, this can be done programmatically. In this paper, we introduce several methods to accurately match food items mentioned in reviews with the official menu item name. These include string-matching algorithms such as `ExactMatch` and `PartialMatch`, as well as a support vector machine matcher, `SVMMatch`. We achieve an F1 score of 0.72 with the SVM and a recall of 0.84 with `PartialMatch` by comparing our matches with annotated results. We also use annotations for sentiment analysis on food mentions and find that most dishes are rated positively by their reviewer. Finally, we do location analyses which reveals that of the seven cities in the data set, reviewers tend to rate Chicago dishes higher.

1 INTRODUCTION

Yelp is generally the go-to site for restaurant recommendations. One can make a decision about a restaurant after scanning through its reviews, looking at the star rating (ranging from 1-5, with 5 being the best), noting the price range on the side bar, and more. But there is a wealth of information buried within this data that is not immediately obvious to a person. A user can read a review recommending a tasty dish, but it is difficult to map this to a particular menu item offered by the restaurant.

A machine can more easily map foods mentioned in reviews to actual menu items, allowing people to know exactly which dishes are popular, tasty, and worth it. This is not a simple task, because reviews often mention food items differently than how they are listed on the menu. For example, a review may rave about the "vegetable and seafood dish", while the menu lists a "chef's seafood delight". Furthermore, colloquial or regional language may be used (e.g. "chips" instead of "fries", or "veggie" instead of "vegetable"), making it even more difficult to make a match. But it is also difficult to *identify* a food mention in the first place!

While an overall restaurant rating is useful, it is also helpful to have per-item ratings. Using human-annotated sentiment analysis, this can be quickly aggregated and presented to a restaurant-goer. Location-based trends in the data can also discovered. For example, New York reviewers may have more expensive tastes and review pricier menu items, whereas reviewers in Boston may like cheaper restaurants but rate menu items more negatively.

1.1 Goal

Our goal is to create several models to precisely identify food mentions and match them up with menu items. We will compare models using the F1 score. This will involve extracting mentions from reviews and annotating the data for a gold matching of mentions to menu items. The annotation tool will be exportable across different platforms and results will be crowdsourced.

Then, using sentiment analysis, we will find a "score" for each menu item mentioned in a review. We then will do several analyses over the sentiments to find interesting trends such as the effect of location on menu item sentiment and the overall average sentiment for menu items.

1.2 Applications

We foresee several applications of our work, all of which can greatly improve the user-experience of anyone using Yelp or other food review sites:

- **Recommendations of popular menu items.** Based on the reviews and number of times a menu item is mentioned, we can figure out the most popular menu items and present this to the user.
- **Favorite menu items.** We can identify how enjoyable specific dishes are using the human-annotated sentiment analysis. This allows us to present a ranking of favorites from a restaurant and have ratings *per-item* instead of for just the restaurant.
- **Search by menu item.** Food mentions within each review and image are tagged, meaning that a user will be able to retrieve the reviews, images, and average sentiment of a specific food item.
- **Location-based suggestions.** Based on review sentiment, we can find averages over different dimensions, such as the average rating of different categories of cuisine. For example, we may find that Boston restaurant-goers rate Chinese dishes more positively, and then suggest highly-rated Chinese dishes to someone looking for food in Boston.

2 RELATED WORK

Some related work has already been done in exploring review data outside of just averaging ratings. We describe two that are most relevant from which we draw our inspiration.

2.1 Consumer Sentiment in Reviews

[1] explores the point that reviews go beyond ratings. Instead, they are narratives that portray reviewers with characteristics or roles such as "victim", “addicted to chocolate”, or "well-educated". A variety of positive and negative reviews are examined. The experiment uses Yelp review data gathered in 2006 to 2011 which we also used.

One method used in their analysis is the “log odds ratio informative Dirichlet prior” [2], which looks for frequency of words in a certain category of the reviews compared to others (e.g. reviews with one star versus five stars or cheap restaurants versus expensive). Another method involves ordered logistic regression to predict a review’s rating score (from one to five stars) or the restaurant’s price (from $ to $$$$). This associates linguistic variables with ratings or price after controlling factors.

The results of this paper indicate the variety of narratives and framings across different kinds of reviews. One star reviews are typically trauma narratives conveying the author as a victim. Positive reviews from cheap restaurants portray the author suffering from junk food cravings. Expensive restaurant reviews use sensual and complex words to show the author’s intelligence and credibility. In general, reviews are positive to positively present their author. This study provides a new methodology to use online text (reviews) and gender computation in order to draw conclusions about the reviewers and the food.

2.2 Product Features and Opinions

In [3], the authors discuss extracting product features and opinions from reviews. It can be difficult for a reader to find relevant information in a review due to differences in what the reviewer and reader value. The authors break down this issue of review mining into four main points: identifying product features (size or speed), identifying opinions regarding product features (“too big”), determining the polarity of opinions (“so great”, “complete disappointment”), and ranking opinions based on their strength (“horrible” is stronger than “bad”).

The paper elaborates on three review mining subtasks. They use OPINE, an unsupervised information extraction system that embodies a solution to each of the subtasks. Given a product and a set of reviews, OPINE returns a set of product features along with a list of opinions ranked by strength. It achieves high precision using relaxation labeling, which finds semantic orientation of words in context.

The second subtask compares OPINE with a review-mining system created by Hu and Liu in 2004. The results showed that OPINE is 22% better in precision but 3% lower in recall for feature extraction. The increase is due to OPINE’s feature assessment mechanism and Web PMI statistics. Finally, most other systems determine the polarity of sentences and documents using extracted opinion phrases. OPINE determines its precision and recall based on “opinion phrase extraction” and “opinion phrase polarity determination” within the context of known product features and sentences.

OPINE’s use of the Web and relaxation-labeling technique allows it to identify product features and customer opinions/polarity with high precision and recall.

3 DATA AND ANNOTATIONS

3.1 Data Set

Our data set is from [1] and includes Yelp reviews from 2006-2011 for restaurants in seven cities: Boston, Chicago, Los Angeles, New York, Philadelphia, San Francisco, and Washington D.C. The data is in JSON format and contains details on each restaurant such as location, category, costs, menu items and their prices, and more.

3.1.1 Data Set Preview

We analyze the raw data to get a better sense of the overall trends and features. We include trends discussed in [1].

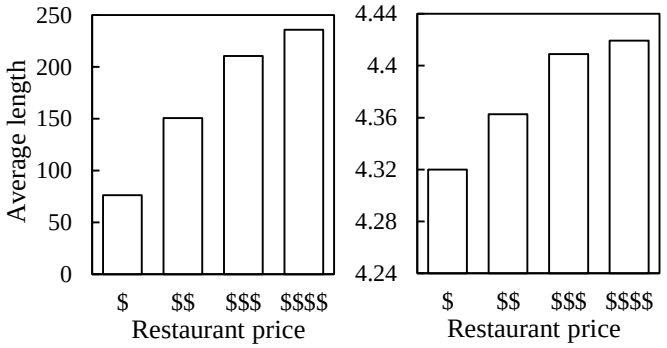

Figure 1: Average length of reviews (left) and words (right) vs. price of restaurant

[1] notes that the longer reviews are associated with more expensive restaurants. Pricier restaurants also have reviews using bigger words. This is possibly due to a correlation of more verbally-intelligent or eloquent people having higher-paying professions, which allows them to frequent more expensive restaurants. Another possibility is that people feel the need to write more about a restaurant if they pay more for it. We see later whether or not more expensive dishes are more enjoyable.

Figure 2: Number of reviews vs. restaurant rating.

From Figure 2, we see that overall restaurant ratings are mostly positive. This may be due to a linguistic history of positive bias in vocabulary. Furthermore, reviewers may only post about a restaurant if they had a good experience; if the restaurant was mediocre or bad, they would not bother writing a review. We later see a similar behavior for per-item sentiment; the average rating of individual menu items is also positive.

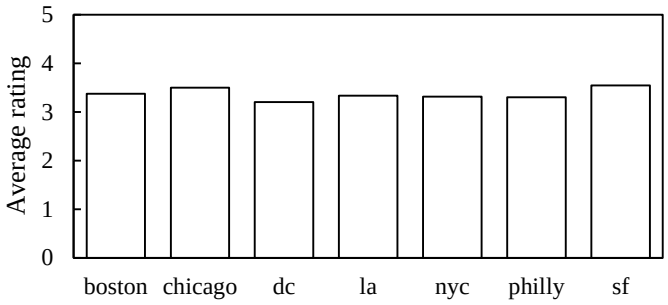

Figure 3: Average rating of restaurants vs. the city.

In Figure 3, we see that the average rating of a restaurant does not vary much by city. We are curious if this changes when we look at it on a per-item basis. We explore this further in Section 6.

3.2 Annotations

3.2.1 Annotation Tool

In order to do sentiment analysis and train and evaluate our models, we need to annotate the data and separate it into `train`, `test`, and `dev` sets. From the 600,000+ reviews from over 5,000 restaurants, we annotated about 3,000 mentions, and set aside 20% each for `test` and `dev`.

To make annotations easier, we created a small annotation program in Python with an interface shown in Figure 4. At first, we considered using Amazon's Mechanical Turk program for mass annotations, but we had doubts about the accuracy of the workers and decided it required too many resources. Instead, we crowdsourced it by enlisting the help of several friends and colleagues who each annotated a different portion of the data set. Each person ran one or more batches of 100 mentions.

```
Text: ≪fried chicken≫
Review: I enjoyed the ≪fried chicken≫ with
buffalo sauce...

0: Not a food item
1: Chef's favorite chicken dish – spicy!
   Sautéed chicken with spicy sauce
2: Chicken sandwich – with dressing & salad
3: Buffalo fried chicken
4: Chicken drumstick – fried
   Two drumsticks plus coleslaw on the side
5: Chicken bake
   Chicken wrapped in Hawaiian rolls
6: Can't tell/ambiguous/not listed above

Which most closely matches ≪fried
chicken≫? Enter the number:

Is the sentiment of ≪fried chicken≫ in
this review:
1: Positive
2: Neutral
3: Negative
Enter the number:
```

Figure 4: Example prompt from our annotator. This includes the context of the mention, possible menu items, and their descriptions. After selecting a menu item match, the annotator is prompted for the review sentiment.

Each run of the tool prompts the annotator to select a menu item matching the mention, if one exists. If a match is made, the annotator is instructed to rate the review as a positive, negative, or neutral description of the menu item. The results of all the runs are combined to form one gold annotated set.

3.2.2 Annotator Agreement

Since the annotations are done in batches, errors will occur. We evaluate the precision and accuracy of our annotators by taking 100 mentions and comparing the matches done by different annotators.

63% of the annotations were exact matches. In 7% of the cases, the difference came from disagreement on whether or not the mention is a menu item. Sentiment analysis only differed in 3% of the annotations, when one annotator thought the review was positive while the other thought it was neutral. The remaining 27% arose when one annotator thought the mention was too ambiguous while the other annotator went ahead and did a menu item match. Since our matching tool treats ambiguous menu item matches the same as a non-food item, we essentially have a 90% rate of agreement among annotators.

We also measure the inter-annotator agreement using Cohen's kappa measure, κ. Using the same data as before, we find $p_o = 0.9$ and $p_e = 0.40$, giving us $\kappa \approx 0.83$.

4 METHOD

4.1 Implementation

4.1.1 Extracting Mentions

Our code is written in Java. We make use of the JSON library to parse the data files and Stanford's CoreNLP library to extract mentions from the reviews. From this, we create a set of `Restaurant` objects, each containing a list of `Reviews`, `MenuItems`, and `Mentions`.

CoreNLP returns *all* mentions found within reviews, even non-food items. We filter these so only food mentions are left, using named-entity recognition ("Miscellaneous", "Organizations", or "People", see Section 4.1.2) and part-of-speech tagging (only noun phrases). Many flagged mentions are numbers, such as food quantities or prices, which we safely ignore.

However, lots of food names are not in English and CoreNLP cannot properly tag them. For example, "Char Siu Pork Bun" is tagged an organization (when it should be a food), and "Tom Yum" is tagged as a person (but actually is a soup). Thus, of the mentions extracted, we keep ones that *could* be foods (all types of nouns, persons, organizations, and things tagged miscellaneous).

4.1.2 Matching Mentions with Menu Items

To test out different ways to match mentions in the `Reviews` with `MenuItems`, we created a `Match` class to be easily extended. This class has a single function that takes in a `Mention`. The `Mention` contains all the information needed to make a match, such as the context it is found in and the possible menu items that it could match. After running, it returns a set of possible menu items to match.

```
public class Match {
  public List<MenuItem> doMatch(Mention m);
}
```

Figure 5: The class for matching mentions to menu items.

We use a total of five different models for matching, discussed further in Section 4.2. Our matching framework is built such that we can easily change the model we want to use as a command-line argument.

4.1.3 Sentiment

After matching a food mention to a menu item, we use human input (via the annotation tool) to figure out the sentiment and determine if a dish is positively, neutrally, or negatively reviewed. For each menu item analyzed, we aggregate over the sentiment from all relevant reviews. However, since the space of menu items is much larger than the space of menu items reviewed and annotated, few aggregations are made.

4.2 Matching Algorithms

We used several algorithms to match mentions with menu items, described in the following sections.

4.2.1 ExactMatch

This matches a mention to a menu item if the mention matches the menu item exactly (ignoring case). This is a very strict model and doesn't make many matches. We chose this to capture the rare occurrence in which a reviewer references the full name of a menu item.

4.2.2 SubstringMatch

`SubstringMatch` will only make a match if every single word in the mention is found in the menu item's name, in the same order. This is a looser requirement than `ExactMatch` since it does not require matching in both directions. This accounts for long menu item names, such as "Grandma Sally's special cheese lasagna", matching with a short mention, "cheese lasagna". This usually finds more than one possible menu item to match with a given mention. We apply disambiguation (see Section 4.3) to reduce it down to a single menu item match.

4.2.3 PartialMatch

`PartialMatch` finds menu items that partially match the mention. A partial match is when more than half of the words in the mention can be found in the menu item's name and description. We added this model after noticing several mentions consisting of many words, some that are extraneous and added by the reviewer. For example, the reviewer might mention "chicken nuggets with barbeque sauce" since they had their food with extra sauce, whereas the menu item is just "chicken nuggets". Furthermore, this accounts for when words are reordered between the mention and the menu item listing.

4.2.4 FuzzyMatch

We noticed that reviews often had typos in their mentions, which would otherwise fully or partially match a menu item. For example, one reviewer talks about "duck friend rice", when they actually meant "duck fried rice". `FuzzyMatch` is less strict about matches by matching a word from a mention to a word from a menu item if their edit distance is less than 3. We use the Damerau-Levenshtein distance, which accounts for insertion, deletion, substitution, and transposition. According to [4], these account for nearly 80% of human errors in spelling, which we want to capture.

4.2.5 SVMMatch

Here we use a support vector machine for learning and classification in order to solve two problems: if a mention is a food item or not and if so, the menu item it matches

with. To do this, we utilize Cornell's `svm_light` tool [6] and our annotated results of the Yelp reviews.

For the first problem, we construct a training file from our `train` data. This file lists all mentions and marks whether or not the mention is categorized as a food item, as well as many bag-of-words features. We create a dictionary of words taken from our annotated mentions' context sentences to create features corresponding to the presence of each word in the mention, before the mention, and after the mention. The `svm_light` tool then creates a model, which predicts on the `test` or `dev` set of mentions. We predict a mention is a food item if the predicted value is greater than 0 (-1 indicating definitely not a food mention, and 1 indicating that it is).

For the second problem of choosing the menu item to match with a food mention, we look through every possible menu item and mark if it is a possible match based off of the annotations. We then use the same bag-of-words features along with four additional features. These include the number of words that overlap between the menu item and food mention, if they are an `ExactMatch`, if they are a `SubstringMatch`, and if they are a `FuzzyMatch` (see Sections 4.2.1 - 4.2.4). This allows us to capture the relationship between a mention and its corresponding menu item and context. After training the model, we use the predictions and pick a menu item for a mention with the highest predicted value.

4.3 Disambiguation

Often, a matcher finds multiple menu items that match a food mention. This makes it difficult to choose a single menu item match and reduces the precision. Thus, we put the choices through filters, which are tried out in order until a single menu item option remains.

4.3.1 Number of Possible Menu Items

We notice that mentions that produce many menu item choices are not food mentions at all (such as mentions that are just the restaurant's name). We toss these mentions and treat them as non-food items. This may miss mentions that *are* food items; however, this occurred more often than not, so we feel it is better to do so.

4.3.2 Number of Matching Words

This counts the number of words that match between the mention and the menu item, with a match in the item name having twice the weight as a match in the item description. While partial matching methods such as `PartialMatch` and `FuzzyMatch` expand the range of potential menu item matches, this reduces it back down for higher precision.

4.3.3 Price

Menu items are priced in an interesting way – not according to the traditional demand curve [5]. That is, pricing is not rational, and unpopular menu items are not necessarily priced lower to entice customers. Most customers have little information about the actual value of a dish, and the only indication is from the price. Thus, an expensive dish is perceived to have higher quality. Someone just trying out a restaurant would not get the most expensive or cheapest item. Therefore, if there are still multiple choices for menu items, we take the median priced one.

4.4 Location-Based Aggregation

We aggregate over a particular location to find average review sentiment, restaurant prices, etc. and reuse most of our code for data-parsing. We put the results through Excel to produce charts. Section 6 describes the results.

5 RESULTS

5.1 Testing Methodology

Our experiment involves trying each matcher, examining the errors that it makes, and the resulting F1 score. We tune the models on `dev`. Training is done on the `train` set. We do not do anything with `test` except evaluate our score at the end.

5.2 Scores

We use precision, recall, and the F1 score to measure the success of each matcher.

Definition. *The precision $\mathcal{P}$ is measured as the fraction of mentions identified as food.*

Definition. *The recall $\mathcal{R}$ is measured as the fraction of food mention to menu item matchings that are correctly identified compared to the gold (annotated) set.*

Definition. *The F1 score is computed as follows:*

$$\mathrm{F1} = 2 \cdot \frac{\mathcal{P} \times \mathcal{R}}{\mathcal{P} + \mathcal{R}}$$

Table 1 below shows the results of each matcher on `test`.

Matcher	Precision	Recall	F1
`ExactMatch`	**0.7951**	0.2945	0.4298
`SubstringMatch`	0.5722	0.7283	0.6409
`PartialMatch`	0.5148	**0.8522**	0.6419
`FuzzyMatch`	0.5212	0.8476	0.6455
`SVMMatch`	0.6659	0.7791	**0.7180**

Table 1. Precision, recall, and F1 scores for each matcher on the `test` data set.

6 EVALUATION AND ERROR ANALYSIS

6.1 Sentiment

In this subsection and in the next we do analysis on the sentiment of the mentioned menu items as identified during annotation.

6.1.1 Overall Average Sentiment

Sentiment is measured on a scale of -1 (negative) to 1 (positive); neutral sentiment has a score of 0. We average the sentiment of the annotated menu items and find an average rating of **0.43**, meaning that most of the menu items received positive sentiment. This is on par with [1] which found that reviews are generally positive, suggesting that menu items mentioned would also be received positively.

6.1.2 Sentiment Based on Price

We also examine how price affects the sentiment of the menu item. Figure 6 shows the average rating of mentions for different price categories.

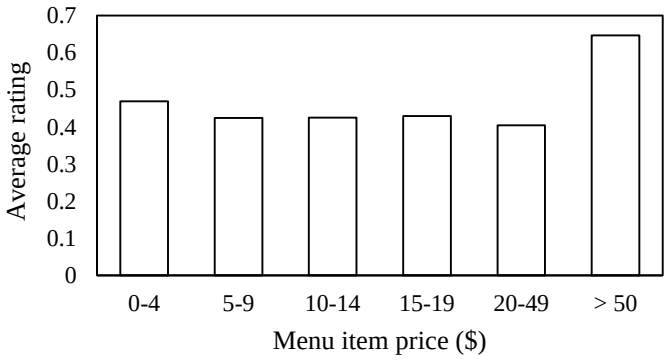

Figure 6: Average rating of menu items for different prices.

We see that the average rating is the about the same for all price ranges except for the most expensive (>$50). We present two possible explanations.

First, it could be due to psychological behavior. Small studies have shown that more expensive food *seems* to taste better. That is, given the same dish, people will rate it higher if they paid more for it. This could be the effect of price on perceived quality [5], or simply because one feels guilty paying more for something and is forced to enjoy it.

Otherwise, this difference may be due to self-selection bias. That is, one would not spend that much money on a dish unless they already knew it was good, thus, inflating the average rating of the menu item.

6.2 Location-Based Analyses

Next, we do analysis by aggregating sentiment and price over the city the restaurant is found in.

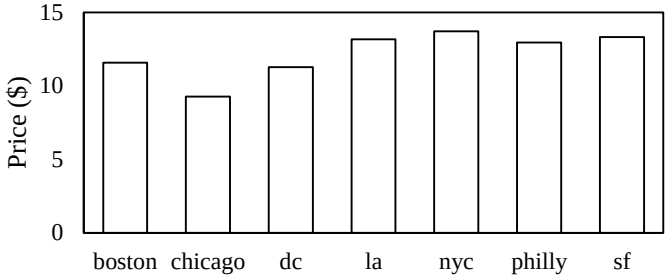

Figure 7: Average price of menu items across cities.

From Figure 7, we see that the average price of mentioned menu items varies between city, with Chicago being the cheapest at $9.25 per item and New York City the most expensive at $13.70 an item.

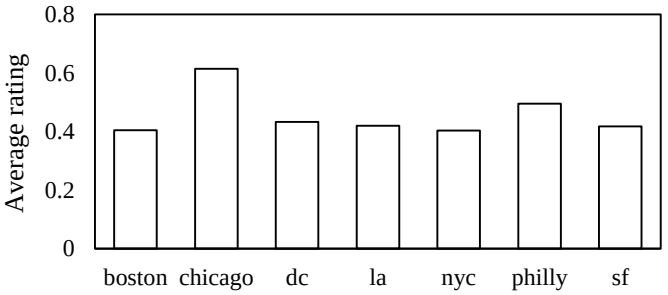

Figure 8: Average rating of menu item aggregated over city.

Figure 8 shows that the average rating of menu items in Chicago is much higher than in other cities, despite its lower average price. We suspect this is due to confounding factors such as the type of food that is popular in Chicago (pizza and hot dogs), but it's interesting that we are able to find a trend based on sentiment analysis alone.

6.3 Errors in the Data

There are several kinds of errors in the data itself that we could not fix, no matter which model we use.

6.3.1 Ambiguous Menu Listings

Some restaurants have menus that are ambiguous, which makes it hard find a single match. Items that come in multiple sizes have separate listings for each size, but nobody mentions the size in their review. For example, soups come in "bowl" and "cup" sizes, and pizzas come in "small", "medium", and "large" sizes. Sometimes, the same menu items were listed twice.

To get around this, we instructed the annotators to always pick the first menu item when matching. Our matchers do the same. This essentially collapses the ambiguous listings into one, but does not fix the problem. We can solve this by pre-processing the menu data to remove the ambiguity; however, this change would make the menus we process different than the menus from the restaurant, which is not ideal.

6.3.2 *Incomplete Yelp Data*

A lot of Yelp data is user-contributed and thus the menu item listing for smaller or less-popular restaurants are sparse. This causes a lot of trouble during annotation and matching, since a food item mention cannot be matched to anything even if it is clearly a menu item. Luckily, this error does not affect our F1 score, since it would not have been hand-annotated either.

6.4 Evaluation of Models

6.4.1 *ExactMatch*

Our `ExactMatch` model, unsurprisingly, does not do very well. Because it requires an exact matching of the mention and the menu item name, it is very cautious when making a guess. For example, it will not match "Frutti Di Mare" with "Frutti Di Mare **Shrimp**" or "Jumbo Lump Crab Cake" with "Jumbo Lump Crab Cake**s**". Most of the errors that occur are due to the menu item having extra words or letters.

We expected 100% precision but were surprised when it only achieved a precision of 0.80. Some of this is due to human error during annotation (discussed in Section 3.2.2) but other cases were due to lack of context. For example, `ExactMatch` matched "Mongolian Beef" with an exact match, but the gold match is "Mongolian Beef Over Rice". We use CoreNLP to find mentions, but it isn't built to find food mentions, so it is unable to expand the mention to capture more of the context (in our example, "Over Rice" is a preposition and not part of the mention).

However, it achieves a recall of 0.29, meaning that about 30% of the time, when a review mentions a food item, it matches the menu item name exactly. This means that the current search functionality on Yelp (which only does exact string searches) misses about 70% of the results! If Yelp implemented one of our inexact string matches (such as those in Section 4.2, like how Google Search works), a lot more results could be captured with a small increase in work.

6.4.2 *Substring, Partial, and FuzzyMatch*

These three matching models were a large improvement over the `ExactMatch`, sacrificing precision for a huge increase in recall. This increase occurs because these models produce a much larger set of possible menu items to match with (`ExactMatch` sometimes gives no possible matches). The F1 score increases slightly between `Substring`, `Partial`, and `FuzzyMatch` as each have a looser criterion for menu item matching.

`SubstringMatch` immediately fixes the problem of cautious matching with `ExactMatch` mentioned above. However, it does not account for word reordering, such as "Teriyaki Chicken" and "Chicken Teriyaki". Thus, we implemented `PartialMatch` to account for this. However, there aren't many instances in which this occurs, as reflected in the minor boost in F1 score between `Substring` and `PartialMatch`.

We used an edit distance of 3 for `FuzzyMatch`, which we found gave us the best F1 score. Decreasing this value made the `FuzzyMatch` operate basically the same as `PartialMatch`, whereas increasing it exponentially increased the number of possible menu items the matcher would choose from. This was so severe in some cases such that every listing on the menu would be provided as options.

The error that seems to occur most often is when the matcher finds a menu item match but the human annotator does not. This happens when the mention is a single word, such as "Mexican" or "New". The matchers, which at most do a substring match, are able to find menu items that have these in their name. However, they do not take into account the context of the mention, which usually reveal that the mention is not a food item at all. We added a filter to immediately toss out mentions with a single word and tried it out with `FuzzyMatch`. Without this filter, our `FuzzyMatch` achieves an F1 score of 0.65, with precision 0.52 and recall 0.85. With this filter, we get an F1 score of 0.66, precision of 0.66, and recall of 0.66. We chose not to keep this filter because while the F1 score increases, the recall decreases. We feel that for a website such as Yelp, false positives are preferable to false negatives.

6.4.3 *SVMMatch*

`SVMMatch` does very well, achieving an F1 score of 0.72. There was an initial concern of overfitting, as we are using a bag-of-words approach for an SVM with over 2000 support vectors and a VC dimension of about 1970. However, this does not seem to be case.

Unlike the previous string matching models, `SVMMatch` matches foods to menu items based on the trained feature weights. Thus, some obvious mistakes occur such as "St. Bernardus" being matched to "Filet Mignon" (a mistake that `PartialMatch` or `SubstringMatch` would not make). Nevertheless, it is cleverer in some sense because it is able to use the mention's context as a feature to help match menu items. Furthermore, we use two SVMs, one specifically targeting the identification of food items, which greatly improves the precision (an increase from about 0.52 to 0.67). For identifying food items, the SVM achieves an accuracy of 84%.

We had the most difficulty getting `SVMMatch` to work. Because `svm_light` only allows features with numerical values, adding mention context increased the number of features by thousands, since we added three features for each unique word. The tool also requires many files, each with a different format. At some point, the runtime exceeded 90 minutes! However, we were able to reduce the runtime to less than 2 minutes after optimizations and error-fixes.

7 CONCLUSION

7.1 Future Work

We had about one month to do the project and our project was scoped accordingly. However, there are many insights to be drawn from the data set that we were not able to do. The following are some of the analyses and experiments we would like to conduct in the future.

7.1.1 Additional Features to SVMMatch

Additional features can be added to `SVMMatch` in order to improve predictions, such as a mention's part of speech in the sentence. We can look for mentions that have similar POS or POS that trends towards nouns and noun phrases, as these are probably more likely to be food items. Another feature could be created based off of the price of the menu item or price range of the restaurant.

7.1.2 Reviewer Gender

Another direction that we would like to look further into is identifying the reviewer's gender from their review and seeing how that affects review sentiment. In related work [1], it was seen that women tended to talk more about dessert. Other than that, there is not much research on differences between gender in reviews. It would also be interesting to see if there are review sentiment differences in accordance to the reviewer's gender.

7.1.3 Restaurant Category

A third thing to look further into is how the restaurant category affects rating and price. We want to see if certain types of cuisine tend to be rated higher or lower and if they have patterns in the pricing.

7.1.4 More Annotations

We had very few annotated mentions during our early testing. As we progressed and accumulated annotations, the F1 scores of our models slightly improved, despite not changing any algorithms. This is due to the small initial sample size; the addition of data only reduced the variance. Adding more annotations will only further improve the data and give us more data for training the SVM.

7.2 Acknowledgements

We would like to thank Vinodkumar Prabhakaran for mentoring us and providing guidance throughout the project. Professor Chris Manning and his teaching staff also provided support. We would also like to thank our friends for helping us annotate the gold data set.

8 REFERENCES

[1] D. Jurafsky, V. Chahuneau, B. R. Routledge and N. A. Smith, "Narrative framing of consumer sentiment in online restaurant reviews," *First Monday,* vol. 19, no. 4, 2014.

[2] B. Monroe, M. Colaresi and K. Quinn, "Fightin' words: Lexical feature selection and evaluation for identifying the content of political conflict," *Political Analysis,* vol. 16, no. 4, pp. 372-403, 2008.

[3] A.-M. Popescu and O. Etzioni, "Extracting product features and opinions from reviews," *HLT Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing,* pp. 339-346, 2005.

[4] F. J. Damerau, "A Technique for Computer Detection and Correction of Spelling Errors," *Communications of the ACM,* vol. 7, no. 3, pp. 171-176, 1964.

[5] J. Carmin and G. X. Norkus, "Pricing strategies for menus: Magic or myth?," *The Cornel Hotel and Restaurant Administration Quarterly,* vol. 31, no. 3, pp. 44-50, 1990.

[6] T. Joachims, "SVMlight Support Vector Machine," Cornell University, 14 August 2008. [Online]. Available: http://svmlight.joachims.org/. [Accessed 21 November 2015].

APPENDIX

You can download all the code and data files needed for this project at the following link: http://anjoola.com/content/cs224n-food.zip. Windows is required to run `SVMMatch`.