Yelp Menu Inference

Jason Jennings **Nicholas Early**

Motivation

Yelp is a great source of information for reviews of businesses, especially restaurants. However, Yelp currently does consistently provide a menu for restaurants. It is up to restaurant owners to provide a menu, which is often provided as a link to an external website. When a link to a menu is not provided, viewers often submit an image of a menu, which is often low quality and not searchable.

In this project, we aim to provide yelp the ability to automatically identify menu items mentioned in yelp reviews. Not only could this provide a menu to users browsing the reviews of a restaurant, but could potentially facilitate other features, such as searching for a restaurant on yelp by menu item.

Personal Objectives

As a semester project, the goal of this project is not to produce ground-breaking research, but rather put to test the skills learned during the course. We searched for a project that would be interesting and potentially useful, but simple enough that it could be implemented in the course of a semester. Additionally, the members of the team wanted experience in the entire process of implementing a classifier, from conception to design, implementation evaluation and deployment.

Data Mining Tasks

The tasks tackled in this project are twofold. The first task is identifying features of a sentence that can be indicators of food, as well as features that are potential foods. The second task is developing a model that can distinguish between foods and non-foods. In the original project description, it was proposed that we may attempt to synthesize duplicate menu items to produce a duplicate-free menu. However, in the interest of time it was decided to focus on producing an accurate classifier, leaving synthesis for future work.

Original Plan

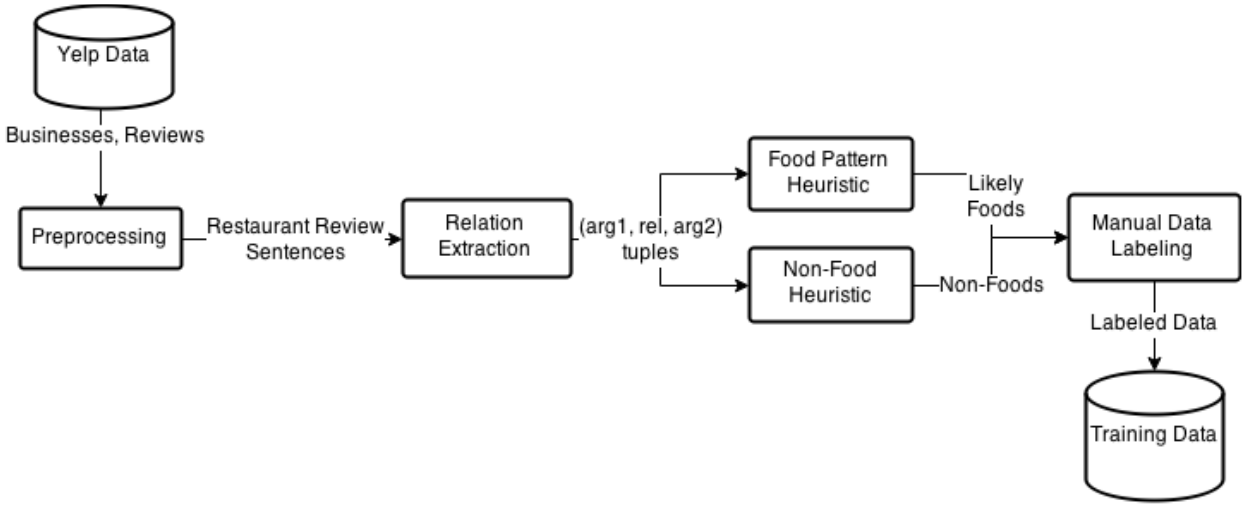
This section outlines the originally planned implementation for this project. It should be noted that as the project evolved, plans were tested and challenges were encountered. As a result, the actual implementation deviated from the plan.

Features
The original project plan involved performing feature extraction using the Stanford Parser, and manually interpreting the syntax trees to extract noun phrases, verbs, and prepositions that correspond to an association. This was pursued but proved difficult. We discovered Open Information Extraction tools, specifically one called ReVerb which essentially performed this task for us, and allowed us to focus on the classification task.

Disambiguation
The original project plan involved a second stage after classification, which would attempt to synthesize named-entities that were likely duplicates of each other. However, because this task had a strong dependence on the task of classification, we decided not to pursue this, and instead focus on implementing a high-quality classifier.

Implementation of Methods

Training Data
The first step of our project is to create a set of training data. The figure below provides a diagrammatic view of how the training data set is generated. The steps will be provided in detail below.



Preprocessing
The data of interest for this project was yelp reviews and yelp business information. This information was contained in two files in JSON format. To facilitate implementation, the JSON files were converted to CSV.

After converting the files to a CSV format, the businesses file is scanned for businesses in the 'Restaurant' category. Then, the set of reviews is joined with the restaurant set to obtain restaurant reviews.

The restaurant review text is further processed and split into individual sentences using NLTK's punkt sentence tokenizer. Punkt implements a sentence tokenizer that is capable of accurately segmenting complex sentences, as well as poorly formed sentences that are common in Yelp reviews, and performs much better than simple strategies, such as splitting on punctuation.

Relation Extraction

The restaurant review sentences are then fed into ReVerb, an 'Open Information Extraction' tool. ReVerb takes sentences and extracts binary relationships from the text of the sentences. Examples of input and output of ReVerb are provided below.

The evaluation of this project in the later sections of this paper does not consider the limitations of ReVerb.

Input Sentence

I went to Subway and ordered a Spicy Chicken Sandwich.

Output

Arg1	Rel	Arg2
I	went to	Subway
I	ordered	a Spicy Chicken Sandwich

Important Note

While the relations produced by ReVerb are fairly high quality, its recall rate is fairly poor, especially when proper grammar and punctuation are not used, which is typical in Yelp reviews.

Additionally, due to time constraints, we only examine the 'arg2' column for potential foods.

Food Pattern Heuristic

To create a large set of labeled training data in a short amount of time, some heuristics were used to select sets of relation tuples that contain a high density of foods. The primary heuristic employed is finding tuples that match the following regular expression: the [A-Z][a-z]+.*?

The intuition here is that a large number of named menu items are referred to in this manner. An example would be the sentence "I ordered the Spicy Tuna Rolls."

'the Spicy Tuna Rolls' matches this regular expression.

This heuristic generated a set of 7000 training examples, which indeed contained a high density of foods.

Non-Food Heuristic

The first heuristic created a moderate sized training set, with a good number of both positive and negative training examples. However, the negative examples all followed a certain pattern, and were not a good representation of all non-foods.

To generate a set of non-foods that better represents all non-food items, the top 100 most commonly occurring phrases were selected from the arg2 column of the ReVerb tuples. The intuition is that the most commonly occurring phrases in these tuples were too frequent to be any specific menu item.

After selection, it was confirmed that this was in fact true, and all of these tuples could be labeled as negative examples. This method generated an enormous set (over 200,000 tuples) labeled as negative.

Manual Data Labeling

Using the tuples identified using the heuristics mentioned in the previous sections, a training set must be created.

First, the set of training examples that matched the **Food Pattern Heuristic** were manually labeled by the members of the group. Certain criteria were discussed in order to be consistent.

Criteria

- A specific menu item must be mentioned (for example “chicken” or “hamburger” is a food, but is not a menu item)
- Drinks are considered, if they are a specific drink

It is worth mentioning that even as humans with some knowledge of cuisine, the members of this group likely do not have the expertise to generate a dataset that is 100% accurate.

The set of training examples selected using the **Non-Food Heuristic** were sampled to augment the training set, but not created a skewed distribution of classes in the training set.

Finally, this data set is saved as the training set for this project.

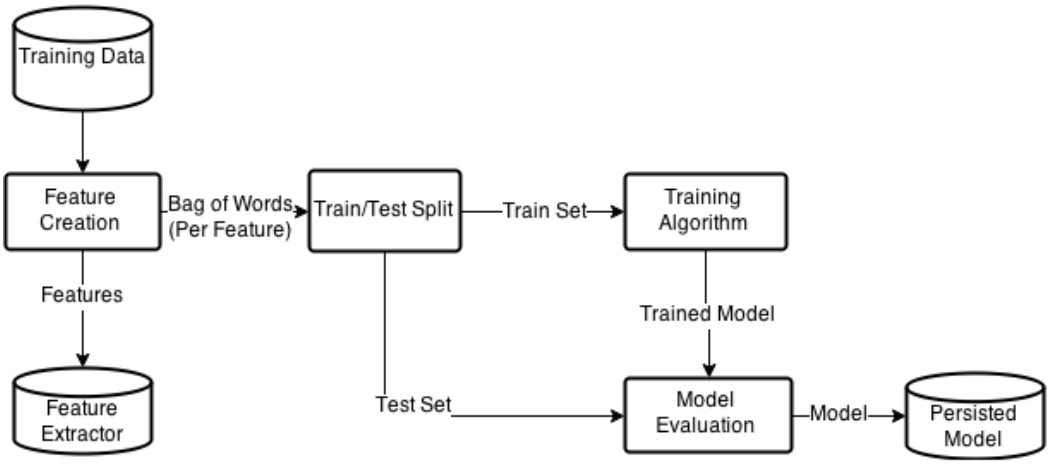
The table below provides an example of training examples in our training set:

Training Examples

arg1	rel	arg2	Label
the strip	love	the Mexican feel and look	0
some great drinks	Try	the Dry Martini	1
They	will about	the Cityscape development	0
I	ca n't not accept	the Gay Egg	1
early They	do not accept	the Port of Subs loyalty card	0
the bacon	Accompanied	the Wedge salad	1
a side of the white wine and dill dressing	usually accompanies	the Lovely Bit of Salmon	1
Luna	had according to	the Yelp reviews	0
this family	achieve	the American dream	0
$ 22	Add	the Huacatay & Roasted Rocoto Sauce	1

Training

After generating a set of training data, the task is to use this data to create a model that can perform well on this data, with the hope that it will generalize to data that it has not seen. The figure below outlines the general process used in this project.



This process was repeated multiple times to generate multiple models, which could then be quantitatively evaluated to select the best model.

Feature Creation

Each feature of the training data is converted to a bag of words model, which is a vector of binary features, describing which words occur in each feature. It is important to note that each feature corresponds to a separate bag of words. The intuition is that a word chicken occurring in the 'rel' feature has a different meaning than the word chicken occurring in the 'arg2' feature.

scikit-learn's CountVectorizer is used to generate the bag of words models. Additional processing is done, such as conversion to lowercase and removal of non-word characters.

Common methods such as TF-IDF, stopword removal, and stemming are not performed. For this task, we believe common words in our data set such as "ate", "ordered", "chicken" are valuable for identifying menu items.

The current implementation of the project generates a very large set of features (a 10995 dimension binary vector). However, these vectors are very sparse and many algorithms in scikit-learn perform well on sparse inputs. Additionally, some models generate weights or feature importance information that can be used to eliminate useless features – this is left for future work.

Finally, the feature extractor is persisted for later use.

Training/Test Split

To produce a more accurate estimate of the accuracy of the model a training/test split is utilized. 75% of the data is used for training, while 25% is withheld to test the model's generalization power.

Training Algorithms

There is no shortage of machine learning algorithms. Due to the limited time and scope of the project, we test only a select few algorithms. Naïve Bayes and Logistic Regression both have a good reputation of being simple, fast, and performing well on text data. Additionally, we tested a Decision Tree classifier, as it is was covered in depth during the course.

Naïve Bayes Classifier

Naïve Bayes classifiers are a well-known machine learning algorithm that is known to perform well on text data. Because our features are Boolean values, the BernoulliNB (Bernoulli Naïve Bayes) class from the scikit-learn package is use. An advantage of Naïve Bayes is it has fewer parameters to tune, and is known to perform well with even with a small amount of data. Additionally, Naïve Bayes not only produces a prediction, but a probability.

Regularized Logistic Regression

Regularized Logistic Regression is another commonly used machine learning algorithm that is known to perform well on text data. It is available in the scikit-learn package using the RidgeClassifier class.

Logistic Regression is a linear model, which can perform well on modestly sized data sets and simple classification tasks. A benefit of Logistic Regression is it produces a weight for each feature. This feature

can be transformed into a log-odds ratio, which can then be interpreted as the 'importance' of that feature. As with Naïve Bayes, Logistic Regression also produces a probability of its prediction.

Regularization refers to a penalty being placed on the magnitude of the weights, with the intuition that smaller weights leads to a simpler model and prevents overfitting.

Decision Tree

Decision Tree classifiers are a simple machine learning algorithm and were covered in depth in the course. Decision Trees have the drawback of having more parameters to tune. Limits may be put on the size of nodes, depth of the tree, different criteria may be used. We tried only a few basic sets of parameters for this project.

Model Evaluation

After training the models described above on the training data, the models are then used to predict the class labels of the test set. The results are then compared to the ground-truth class labels for the test data and evaluated on accuracy, precision, recall, and the f1-score performance metrics. A table outlining the performance of different models is shown below.

Quantitative Analysis

Classifier Name	Accuracy	Precision	Recall	F1
Ridge Regression	**0.918**	0.919	**0.872**	**0.895**
Naive Bayes	0.904	**0.936**	0.817	0.873
Decision Tree 15*	0.825	0.834	0.743	0.790
Decision Tree	0.864	0.844	0.812	0.828

While several of the models perform similarly, the best all-around model (based on the F1 score) is the Ridge Regression model.

However, for this task we consider precision of higher priority. With a large number of reviews, the recall rate can be overcome. A model with lower precision will lead to more incorrect items appearing in the menu. For this reason, we select the Naïve Bayes classifier as the best model.

The Decision Tree classifiers were not competitive with the other models. Perhaps with more parameter tuning, or utilizing ensemble methods could improve their performance.

Because of the heuristics used to generate the training data, the training set and test set is likely not a good representation of the actual distribution of the yelp restaurant reviews. As such, the metrics described in the previous section are considered optimistic performance measures.

Additionally, all performance measures are calculated AFTER ReVerb has extracted binary associations from the data. We do not have ground truth data for raw sentences. Thus, the recall of ReVerb has a significant impact on the actual end-to-end performance.

Qualitative Analysis

To give a taste of how the system performs 'in the wild', a few examples are provided below. These are reviews taken from Yelp pages of local restaurants.

Review Text: *"I love this place! Everything is so tasty... from the breakfast to the cocktails. We eat here at least 2 times a week. I highly recommend the Chicken Chilaquiles with latin potatoes and refried beans, Grilled Shrimp Salad, the Taco platter is greatness too, just to name a few. The thing that keeps me coming back other than the wonderful food is that you can mix amd match, customize the menu to your taste and preference. That is huge for me considering I am gluten intolerant. Thanks, Fuzzy's!"*

Extracted Menu Items: the Chicken Chilaquiles, Shrimp Salad

Comments: While both extracted menu items appear reasonable, the "Shrimp Salad" was actually "Grilled Shrimp Salad". Additionally, "the Taco Platter" is a menu item mentioned in the review that was not identified (in this case, the ReVerb system did not generate a tuple containing this menu item).

Review Text: *"If you want a really good burger and a nice cold beer this is the place for you. We come here about once a month and the burgers are some of the best I've had. This little burger joint / sports bar / dive bar is two separate business in one. One side of the place you order you food and the other side is the bar / eating area. I hear you can order food at the bar but I never have. I normally get the Roasted Red Pepper Burger or Fried Portobello Mushroom Burger. My other half normally has the Angry Burger (sometime spicy and other times it will set you on fire). Recently I had the Chicken Fried Burger which, to my surprise, was very very good. The homemade fries are good but nothing to write home about. The onion rings used to be extremely good but they have changed them and now they are below average. The burgers are big enough so split an order of fries they are plenty big enough for two. The bar is a full bar. I normally just get the beer. They have a wide assortment of Craft beers on tap and in bottles. Good food and good beer, what more could you ask for."*

Extracted Menu Items: the Roasted Red Pepper Burger, the Angry Burger, the Chicken Fried Burger

Comments: Again, the extracted menu items seem reasonable. In this case, "The Fried Portobello Mushroom Burger" was not identified. Again, ReVerb did not generate a tuple containing this menu item.

Future Work

This section outlines steps that may be taken in the future to improve upon this project

Relation Extraction

This project relied on ReVerb for performing relation extraction. Many errors made by the system were as a result of relations not identified by ReVerb. ReVerb is a general purpose tool for extraction relations, attempts could be made for more targeted relation extractions for the domain of this project (Menu Items).

Named Entity Recognition

This project used simple classifiers to perform what is essentially a Named Entity Recognition task. There are open source named entity recognition tools in the Stanford NLP suite, as well as Apache's OpenNLP. The performance of the approach of this project should be compared with the more sophisticated approach of these systems.

Synthesis/Disambiguation

If this system was actually implemented on Yelp or similar websites, it is likely that it would generate multiple duplicate extractions for the same menu item, with slight differences. Investigating the best way to identify and eliminate duplicates is left for future work.

Website

We provide a web-application that allows a user to interactively enter reviews and have those reviews run through the entire process of extracting relations, and classifying the relations as food or non-food. Some examples from reviews of local restaurants are provided, but anything may be entered.

Implementation Details

The website is implemented using Flask, a lightweight web application framework for python. The website communicates with ReVerb via interprocess communication, and this communication is currently somewhat inefficient (a new process must be spawned for each submission).

References

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

: Cambridge University Press, 2008.

Introduction to data mining. Tan, Michael Steinbach, and Vipin Kumar. Boston: Pearson Addison Wesley, 2005.

"Pandas: Powerful Python Data Analysis Toolkit." Web. 04 May 2015.

"Flask Documentation." http://flask.pocoo.org/Web. 04 May 2015.

Identifying Relations for Open Information Extraction, Fader et al.

Reverb – Open Information Extraction Software. http://reverb.cs.washington.edu/

Convert the Yelp Academic dataset from JSON to CSV files with Pandas, Paul Butler. https://gist.github.com/paulgb/5265767

Natural Language Processing with Python, Bird, Steven, Edward Loper and Ewan Klein (2009). O'Reilly Media Inc.

Yelp Dataset Challenge. Yelp Inc. http://www.yelp.com/dataset_challenge