Sentiment Analysis on Food Items in Yelp Reviews

Sorathan (Tum) Chaturapruek, Raphael Townshend
{sorathan, raphtown}@cs.stanford.edu

December 8, 2014

1 Introduction

The majority of people today consult online reviews on Yelp before they decide which restaurant to go to. However, it is not easy to compare a specific food item across multiple regions. Do the four stars in each same place really mean the same thing? A user could read all the reviews for each restaurant candidate to get more ideas about each place, but this process usually takes too long for practical purposes.

Given a food item on Yelp (e.g. burger), we want to score, classify, and visualize various food attributes according to reviews and compare the same food item across various restaurants or regions, our hypothesis being that good reviews of specific food are geographically linked (e.g. lobsters in Maine are much better than other places). We want to summarize sentiment that people feel about a specific item as well.

2 Overview of Our Approach

We begin by classifying the sentiment in every sentence that contains food items of interest (using SenticNet, Sentic API – `http://sentic.net/api/`) and annotating about 1200 samples of the data for evaluation and learning. We want to predict sentiment polarity score given a review sentence. Similarly to assignment 3, we approach this problem in two ways: rule-based approach and featured-based approach.

The visualization shows the polarity broken down by every place, and we support it with snippets found on Yelp.

2.1 Definition

Sentiment. In our study, sentiment means sentiment towards a specified food item. It does not measure the absolute quality of the food but instead measure the difference between outcomes and expectations of users. For example, if a user says, "the burger was quite good, but I was disappointed because I expected it to be the best in the world," we will consider this snippet as negative sentiment.

2.2 Tools and Related Work

SenticNet's sentiment polarity database and concept parser. See [3]. For example, the phrase `i was sick for 2 days after i eat one of their burgers` has the concepts `sick_for_day`, `eat_one`, and `one_of_burger`.

NLTK's POS Tagger. We use NLTk tools [2] to do POS tagging for each word in each sentence.

Malt Dependency Tree Parser. We use the Malt parser [5] to parse a sentence into a dependency tree. We believe that the distance of words in the dependency tree more accurately represent a connection between words in the sentence and the word of the target food item. The parser was trained on the Penn Treebank and then applied to the phrases in our labeled sets.

Tum's previous visualization work. Tum used to work on visualization of four dimensions of sentiment on Google map.[1] In this project, we added some more work to this visualization tool to allow us to view sentiment across multiple regions in the map.

[1] `https://github.com/pbhuss/Sentimental`

Previous NLP final project work Gupta et al. (2009)[2] have done similar work but primarily focus on catogorizing each word in the review into `food`, `ambiance`, and `service` categories.

2.3 Data

The Yelp Challenge Dataset[3] includes data from Phoenix, Las Vegas, Madison, Waterloo and Edinburgh, having more than 1M reviews and 42k businesses. We extracted around 800 reviews (mostly from Madison and Phoenix) that have sentences that contain food items of interest. In our case, we picked the following four food items to study: beer, lobster, burrito, and burger. We picked these four to represent some diversity in food (drink, exotic food, ethnic food, and fast food, etc.).

2.4 Gold Data

The gold data serves three purposes. First, it is a ground for evaluation. We can evaluate how our sentiment prediction algorithm performs against this gold data. Second, for a learning-based algorithm, we can use part of gold data to make the algorithm learn from data and use the rest of the gold data as a test set. Finally, when we label gold data, each data sample gives us intuition about what makes a review snippet have positive and negative sentiment, and shows us various requirements that any simple algorithm cannot predict the sentiment of the given snippet (e.g., sometimes the algorithm needs to have world knowledge to be able to correctly interpret the review snippet. See our Future Work section for more detail).

To make a gold dataset, we assigned a polarity score from $\{-2, -1, 0, 1, 2\}$ to each sample entry, which is a sentence that contains the word of interest. A score of 2 means very positive sentiment (e.g., `the best burger in the world`), whereas a score of -2 means very negative sentiment (e.g. `the worst burger I had in my life`).

	#labeled samples	#unique reviews	$\#-2$	$\#-1$	#0	#1	#2
Beer	289	200	0	19	76	190	4
Lobster	284	200	0	30	122	132	0
Burrito	291	200	0	31	111	149	0
Burger	340	200	4	62	97	155	22

Table 1: Initial manual labeling results.

	#labeled samples	#unique reviews	$\#-1$	#0	#1
Beer	289	200	19	76	194
Lobster	284	200	30	122	132
Burrito	291	200	31	111	149
Burger	340	200	66	97	177

Table 2: Compressed labeling results.

Initially, we used the numbers ± 2 to mean that sentiment is extreme. However, sometimes it is difficult even for human to distinguish between 1 and 2, so we collapsed both of them into 1 and collape -2 and -1 to -1. Thus our revised gold data statistics look like that in Table 2.

2.5 Error Metric

We used the 0-1 loss function. This means we simply counted how many samples that we correctly predict their categories. Even though a sentiment score could be interpret as a continuous function between -1 and 1, we think it was too arbitrary to distinguish sentiment to a detail finer than the three values -1, 0 and 1.

[2] `http://nlp.stanford.edu/courses/cs224n/2009/fp/9.pdf`

[3] `http://www.yelp.com/dataset_challenge/`

2.6 Data Preprocessing and Score Approximation

Given a food item (e.g., `burger`), we searched for all reviews that contain that word and tokenized into sentences using a period. For each sentence, we tokenized into words with NLTK and also used NLTK to obtain POS tagging. We stemmed each word that ends with `['ing', 'ly', 'ed', 'ious', 'ies', 'ive', 'es', 's', 'ment']`. Then, when we looked up a polarity score from the Sentic sentiment database. If the word was not in the database, we took the average of sentiment of words that contains our word of interest. For example, the word `force` itself is in neither the SenticNet database nor the concept list. When we looked up concepts in the concept list that contain `force`, we found `force_other_person`, which has a polarity score of -0.075, and `arm_force`, which has a polarity score of -0.035, so we took their average and assign a score of -0.055 to `force`.

3 Rule-Based Sentiment Polarity Scorers

Our first approach was using a rule-based sentiment scorer. The SenticNet database contains a polarity score of a specific word. Our task was to find a way to meaningfully combine these scores. For example, we incorporated a rule to deal with negation. For example, the sentence `I do not like this burger` clearly has negative sentiment, but we if looked up polarity score of each word separately, we would get a positive score because `like` has a positive polarity score. When we did an error analysis on our results, we found more words to include in the negation category:

("except", "but", "not", "than", "no", "never")

3.1 Concept-Based Scorer

We first employed the SenticNet concept extractor on the phrases extracted from the Yelp reviews. These yield high-level concepts which we can then feed through SenticNet's polarity rating system. For example, the phrase `i was sick for 2 days after i eat one of their burgers` would have the concepts `sick_for_day`, `eat_one`, and `one_of_burger`. If the full concept is not present in the database, sub-phrases are taken until appropriate entries are found. For example, `sick_for_day`, which is not in the SenticNet database, could be broken down into `sick`, `for`, `day`. The sub-phrasing technique is also used in the lookup stage, where if a concept is not present, its value is computed from the average of all other concepts that contain it. Continuing with the same example, `day` is not directly present in the database, but concepts like `day_off` are. These "container" concepts' polarity are looked up and averaged.

3.2 Adjective-Based Scorer

The concept-based scorer was heavily based off of the SenticNet concept parser, which yielded very sparse results. Our next attempt at implementing a scorer involved using the NLTK toolkit to POS tag the phrases. Using these tags, we then ran all adjectives through the polarity scorer and averaged the resulting scores. This yielded a much denser set of lookups and generated a more reliable score.

3.3 Dependency-Parsing-Based Scorer

We next attempted to improve the adjective-based scorer via the use of dependency parsing. MaltParser, a data-driven dependency parser, was used. The parser was trained on the Penn Treebank and then applied to the phrases in our labeled sets. The food word of interest was then located within the parse tree, and then a traversal was attempted from there (very much in the style of the Hobbs' algorithm). First, adjectives dependent on the food word had their polarity evaluated, as they are one of the main sources of information regarding user opinion on the food. Then, the tree was traversed upwards, going towards the root. If any verbs or nouns were traversed on this path, their polarity score was also computed, the reasoning being that there is a one-way dependency between them which implies correlation. If any negation words were found as a dependent of a verb in the traversal, then the current polarity was was inverted. For example, see Listing 1. We start moving up the dependency part from the word `burgers`. We first traverse the word `like` which adds a positive polarity rating. However, when we traverse the verb `did` we see that one of its dependents is the negation word `not`, thereby reversing the polarity of the word `like`.

Listing 1: Inversion Example

```
Phrase: I did not like the burgers
(did
    i
    not
    (like
        (burgers the)))
```

3.4 Important Implementation Decisions

(a) The computation time bottlenecks were when we called NLTK to do POS tagging, call SenticNet to get sentiment scores, called the concept parser to get a list of concepts, and called the Malt parser to get dependency trees. To remedy this issue, we used a cache system to make things a lot faster.

(b) We stored everything in JSON to make it easier to output to something compatible with the front end.

(c) We used a retry mechanism to call external tools, since we relied on calling API across networks.

(d) For front-end visualization, we use AngularJS and jQuery, together with Google Map API. One of the unexpected implementation detail was that we needed to use a closure to be able to make a set of markers together with an information window and still be able to link the right information window.

3.5 Results

We report results on all 3 scores in Figure 3, showing the mean predicted value along with standard deviation. The scores all normalized within each scorer so that they can be comparable across scorers. We can see that both the adjective scorer and dependency scorer outperform the concept scorer (e.g. more spread out means, lower standard deviations) due to the latter's very sparse nature. The dependency scorer also better spreads out the neutral and negative examples, most likely because the neutral phrases tend to contain negative words but not specifically about the food item in question, a case the adjective scorer would fail on.

	Concept Scorer	Adjective Scorer	Dependency Scorer
negative	-0.157 ± 0.910	-0.333 ± 0.878	-0.280 ± 0.866
neutral	-0.227 ± 0.661	-0.339 ± 0.576	-0.246 ± 0.578
positive	0.176 ± 1.149	0.286 ± 1.135	0.216 ± 1.170

Table 3: Scorer results, reported as mean ± standard deviation.

3.6 Error Analysis

We shall now proceed to do some error on the dependency parser. A common error seen was the failure to account for words which were not directly on the parse path. For example, see Listing 2. We would like to consider the adverb `burnt` for rating burger, but we would have to traverse downwards from the verb `was` to reach it. Blind traversal of all descendants would bring us back to the adjective parser and thus we would need some selective way of deciding when to traverse downwards and how far to traverse. Again, this reminds us of the pronomial anaphore resolution problem and its rule-based solutions.

Listing 2: Dependent Failure

```
Phrase: medium-well burger was burnt
(was
    (burger medium-well)
    burnt)
```

In the next example (Listing 3), we see two issues. For one, the dependency parse failed, which quite evidently can cause issues. However, the other problem is caused by the various forms words can take. We cross the word `amazing` on our traversal, but the problem is that it is the word `amaze` that is in SenticNet. We partially solved these by using stemming (trimming the suffixes), but we still fall short here because we would then need to append an e. Knowing when to apply stemming/addition is tricky and would require a more in-depth understanding of the source words (or perhaps a learning algorithm).

Listing 3: Parse and Stemming Failure

```
Phrase: a total dive but the burgers are seriously amazing
(dive
    a
    total
    but
    (amazing (burgers the) are seriously))
```

A final and tricky issue can be seen in Listing 4. Idioms are difficult to capture in the dependency tracing framework because they require the simultaneous consideration of multiple words. In this example, the positive idiom `stop by again` is missed because only the word `stop` is considered in the upwards traversal and it is has a negative polarity. This is exactly the sort of problem the concept scorer was supposed to address but the problem is its sparseness prevents it from being very useful.

Listing 4: Idiom Failure

```
Phrase: but I would stop by again for the gritty burger
(stop
    but
    i
    would
    (by again)
    (for (burger the gritty)))
```

4 Feature-Based Sentiment Classifier

We next worked on training a feature-based classifier three classes corresponding to positive, neutral, and negative sentiment.

4.1 Features

(a) The output of the 3 rule-based scorers are used as features for the classification. The idea is that they capture different properties of the data. As mentioned in the error analysis, the concept scorer is better at representing higher order ideas, whereas the adjective scoring captures more fine-grained polarity, and the dependency scorer even more so.

(b) We also use pleasantness, aptitude, sensitivity, and attention scores as features.

4.2 Model

The model used was a support vector machine. The scikit-learn python package was used for this purpose. There were 280 phrases used per food type, with 140 of them being used for testing, and 140 used for testing. 5-fold cross-validation and grid search was used for model selection, varying the kernel between linear, polynomial, and RBF. The various parameters for these kernels (C, coefficient, gamma) were also determined via the grid search procedure.

4.3 Results

Table 4: Classification results on beer.

Beer	Support	Baseline (all 1's)			Our Predictor		
		precision	recall	f_1 score	precision	recall	f_1 score
−1	7	0.00	0.00	0.00	0.25	0.14	0.18
0	34	0.00	0.00	0.00	0.59	0.38	0.46
1	99	0.71	1.00	0.87	0.77	0.89	0.83
avg/total	140	0.50	0.71	0.59	0.70	0.73	0.71

Table 5: Classification results on lobster.

Lobster	Support	Baseline (all 1's)			Our Predictor		
		precision	recall	f_1 score	precision	recall	f_1 score
−1	17	0.00	0.00	0.00	0.25	0.06	0.10
0	54	0.00	0.00	0.00	0.46	0.83	0.60
1	69	0.49	1.00	0.66	0.64	0.36	0.46
avg/total	140	0.24	0.49	0.33	0.53	0.51	0.47

Table 6: Classification results on burrito.

Burrito	Support	Baseline (all 1's)			Our Predictor		
		precision	recall	f_1 score	precision	recall	f_1 score
−1	16	0.00	0.00	0.00	0.00	0.00	0.00
0	46	0.00	0.00	0.00	0.41	0.78	0.54
1	78	0.56	1.00	0.72	0.79	0.53	0.63
avg/total	140	0.31	0.56	0.40	0.57	0.55	0.53

Table 7: Classification results on burger.

Burger	Support	Baseline (all 1's)			Our Predictor		
		precision	recall	f_1 score	precision	recall	f_1 score
−1	22	0.00	0.00	0.00	0.38	0.14	0.20
0	36	0.00	0.00	0.00	0.34	0.31	0.32
1	82	0.59	1.00	0.74	0.63	0.77	0.69
avg/total	140	0.34	0.59	0.43	0.52	0.55	0.52

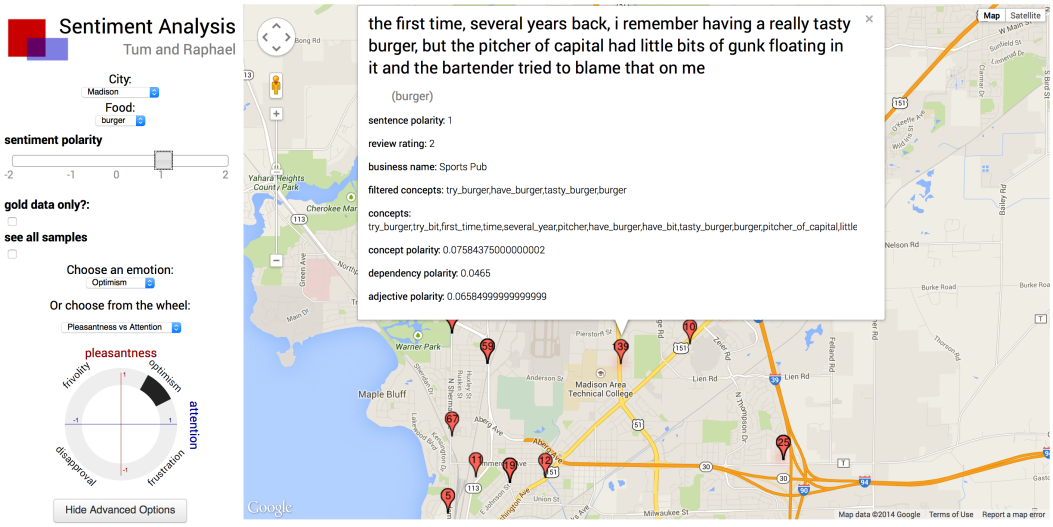
Note that feaures such as pleasantness and aptitude scores are important to differentiate reviews, because otherwise we would encounter issues in which we predicted all 1's.

5 Visualization

We store each data entry as a Python dictionary that has the following keys: sentence rating (either by hand or by our algorithm), latitude and longitude, business name, and other metadata. We then use these data to plot on a map using Google Map API. Each data entry is assigned a marker. When it is clicked, an info window will show up. See an example below.

5.1 Polarity Visualization

Below is an example of polarity visualization. On the left panel we can select the range of polarity score, and we use an epsilon ball centered at the slider (where $\epsilon = 0.4$) to filter results to keep ones that have polarity scores close to the specified score. We also rank them by the polarity score (a low rank number on the marker means a higher polarity score). We provide an option to show gold data only and an option to see all data samples, regardless of their polarity scores. There is also an option to hide all markers (not shown in this figure) to be able to see the heatmap in the section more clearly.



5.2 Hourglass of Emotions Visualization

According to Cambria et al. [4], in the model called the hourglass of emotions, there are four axes of dimensions of emotions: pleasantness (p), aptitude (ap), attention (at), and sensitivity (s). Other emotions can be expressed as a combination of these four emotions. For example, positive pleasantness and positive aptitude are combined to "love", while negative pleasantness and positive aptitude are combined to "envy".

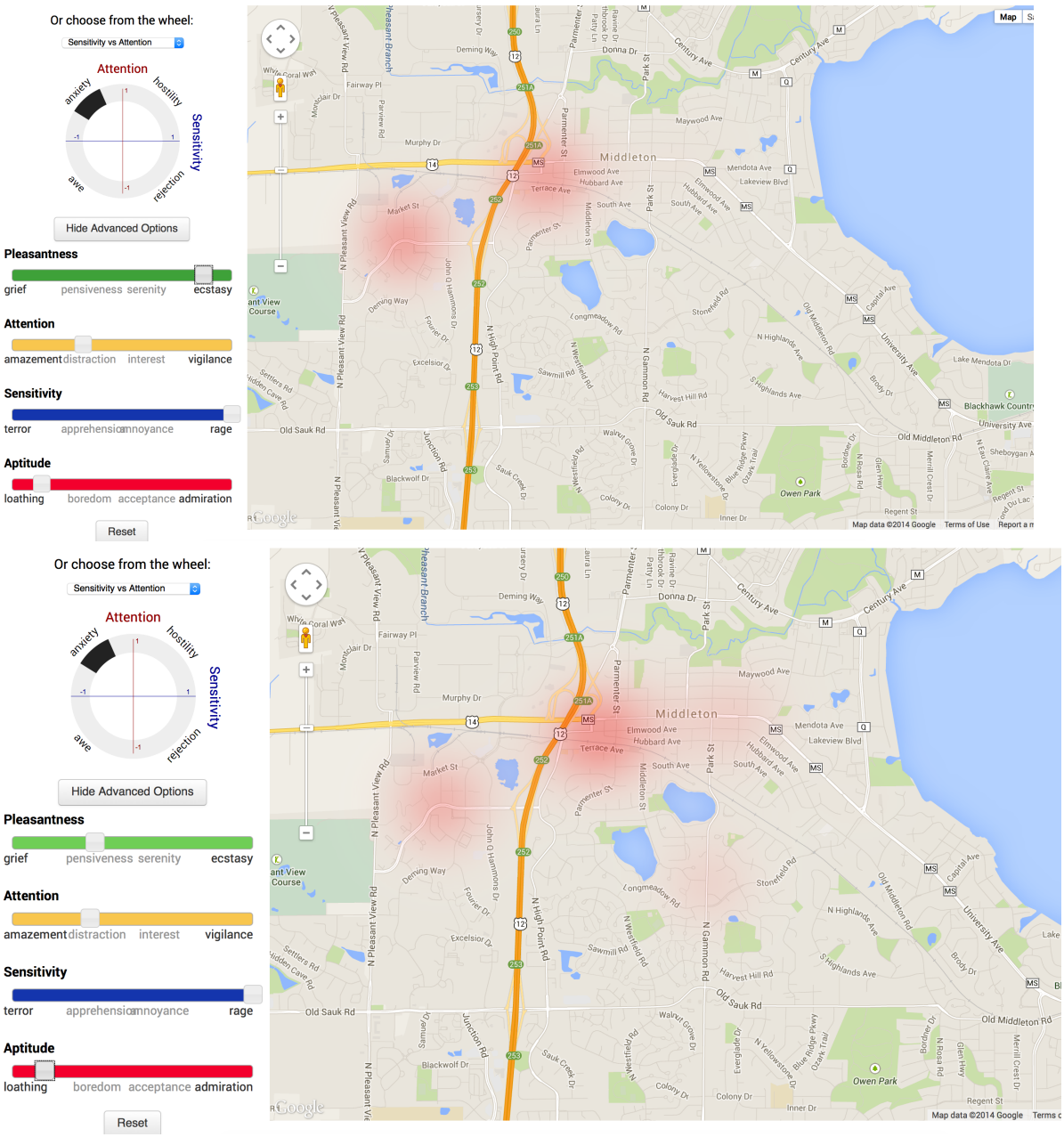
For each word in a sentence, we run through SenticNet database to get these four numbers. We take the average of each score across tokenized words in a sentence when such word exists in the SenticNet database. We then store these numbers and use as features in our prediction.

To visualize these numbers, we devise a metric to capture the notion of closeness between two parameter sets:

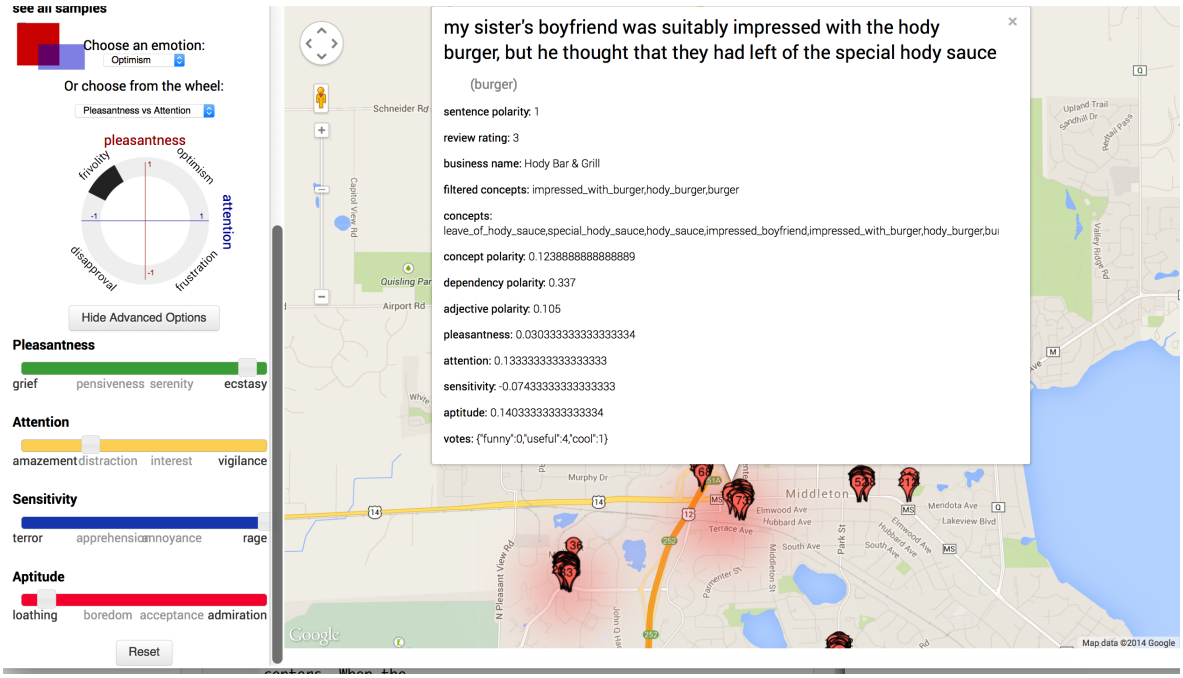
$$weight((p_1, ap_1, at_1, s_1), (p_2, ap_2, at_2, s_2)) = \frac{1}{\sqrt{(\Delta p)^2 + (\Delta ap)^2 + (\Delta at)^2 + (\Delta s)^2 + \epsilon}},$$

where ϵ is a small number to avoid the weight going to infinity. If the weight is high, it means two parameter sets are close. We have a panel of parameters on the left and then the heatmap of restaurant reviews have the four parameters are similar to our setting, they will get higher scores. For example, if we tune my parameter setting on the left to have a very high pleasantness, the heatmap that we get will concentrate on an area with high pleasantness.

Below is an example of such heatmap. The first one shows a setting with a high pleasantness and there seem to be two centers. When the pleasantness is increased we see a different hotspot pattern.



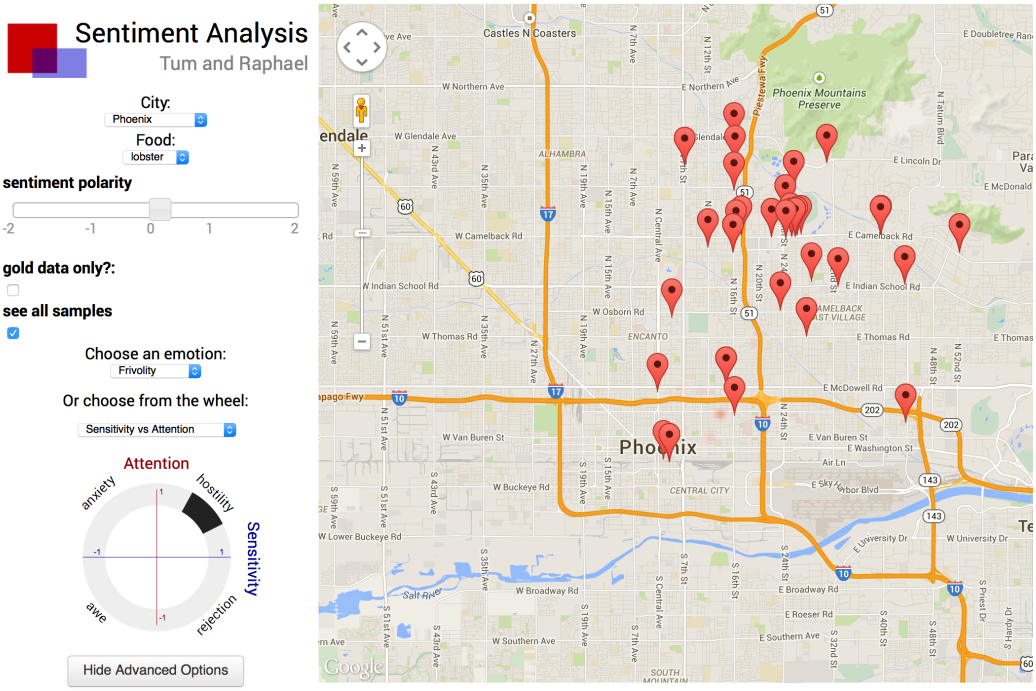
We can show markers and trace which restaurants that correspond to these hotspots.

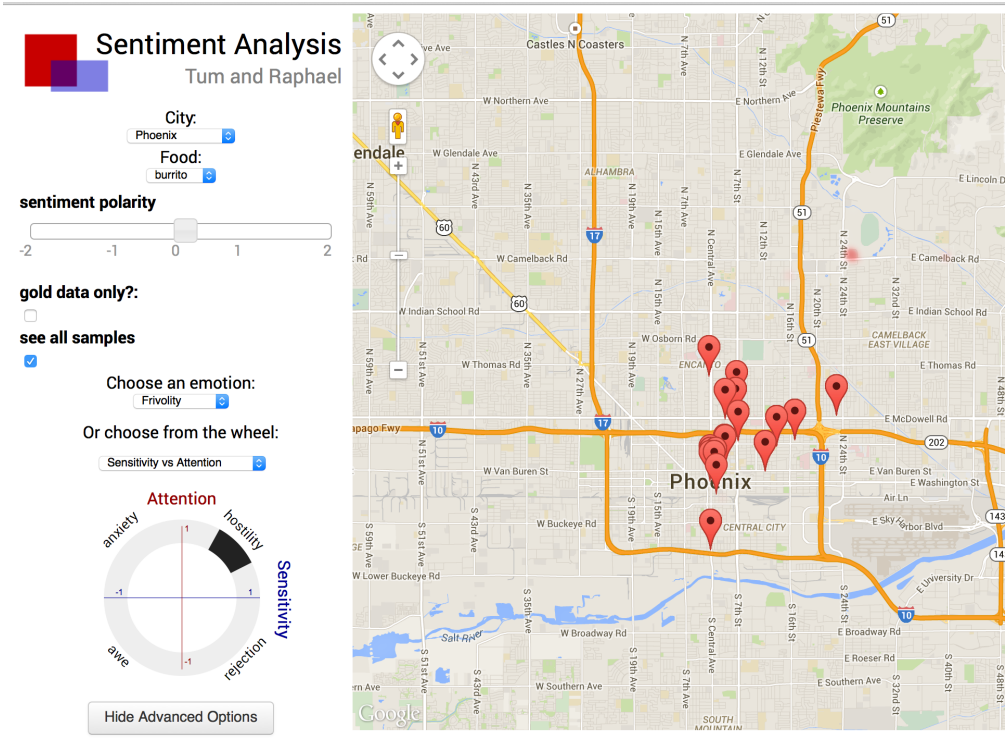


5.3 Trends

5.3.1 Lobster versus Burrito in Phoenix

The following example shows a clear difference between the distributions of reviews that contain the words burrito and lobster, respectively, in Phoenix.





5.3.2 Beer is good!

Based on the small labeled data that we had, beer has the highest percentage of positive sentiment (67% are positive and 26% are neutral, while only 7% are negative). See Table 8 for more quantitative values. This table is derived from Table 2.

	#labeled samples	−1-label percentage	0-label percentage	1-label percentage
Beer	289	7%	26%	67%
Lobster	284	11%	43%	46%
Burrito	291	11%	38%	51%
Burger	340	19%	29%	52%

Table 8: Percentage of sentiment polarity.

6 Future Work

6.1 Incorporating World Knowledge

A system that has world knowledge has a better chance to perform well in the following scenario:

Example: "a big mac would probably be as flavorful as the burgers served here"

Explanation: Our system does not know about a big mac, so it does not know the correct semantics of a review when the review is being compared to a big mac.

6.2 Improving POS Tagger

Currently we tokenize a sentence into a set of words and then use nltk to get a POS tag for each of the words. Sometimes this does not perform well because a word by itself is inherently ambiguous. For example, `play` could be

a noun or a verb. For future work, we could try to use more context to disambiguate between POS tag candidates.

6.3 Improving the Dependency Tree Parser

We did an extensive error analysis of how dependency tree could be improved, and such improvement will have a direct impact on our sentiment polarity predictor.

References

[1] Sentiment based summarization of restaurant reviews final project report CS 224N. `http://nlp.stanford.edu/courses/cs224n/2009/fp/9.pdf`. Accessed: 2014-12-07.

[2] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[3] E Cambria, D Olsher, and D Rajagopal. Senticnet 3: A common and common-sense knowledge base for cognition-driven sentiment analysis. In *Proceedings of AAAI, Quebec City*, pages 1515–1521. AAAI, 2014.

[4] Erik Cambria, Andrew Livingstone, and Amir Hussain. The hourglass of emotions. In *Proceedings of the 2011 International Conference on Cognitive Behavioural Systems*, COST'11, pages 144–157, Berlin, Heidelberg, 2012. Springer-Verlag.

[5] Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *In Proc. of LREC-2006*, pages 2216–2219, 2006.