Multi-Aspect Sentiment Classification of Reviews Via Sentence Selection

Background and Problem Statement

The data we worked on are cruise reviews from the website cruiseCritic.com. Customers write text reviews for cruises they have taken, and give ratings out of 5 for 10 different aspects of the cruise (see figure below).

![Ratings and extract from text content of a cruise rating on cruisecritic.com](image-url)

Background and Problem Statement

The data we worked on are cruise reviews from the website cruiseCritic.com. Customers write text reviews for cruises they have taken, and give ratings out of 5 for 10 different aspects of the cruise (see figure below).
Our first idea of a task was to predict each of the ratings based on the text of the reviews. For this, we wanted to extract the pieces of text that are relevant for each rating and see what performance we could achieve.

An additional idea came from search results from Bing. For restaurants for instance, Bing gives review results from different websites, and organizes the information in a ScoreCard:

From these different sources of inspiration, our goals are the following:

1) On a cruise, you spend most of your time on the ship, therefore obtaining information about different aspects is crucial to the customer. We want to present snippets of reviews to customers, classified by subject rating. Reviews are usually long, and related to personal events (some people post their day-to-day travel logbook), and they are not always organized. Selecting pieces of text by subject will help customers get the specific information they want. It would solve a big problem of peer reviews: users usually trust them more than traditional information websites, but they are messier.

2) Some reviewers do not rate all categories. We can try predicting the ratings for these. This will help complete the data, but also will act as an indicator of the quality of our text filtering: if the filtered sentences are relevant, the classification will be better.

These two classification tasks (classification of rating based on content and classification of sentence category) are what we will focus on for the remainder of the paper.

System Design

Our system is designed around the idea of filtering and weighting the data on which the classifiers are trained. Ideally, we would want to determine the exact set of

sentences relevant to the topic and train based only on our sentences. Below, we first describe our approach to sentence filtering, which is based on learning a set of keywords derived from a set of hand-generated seeds. Next, we describe the implementation of a Naive Bayes classifier that uses the filtered data to predict ratings. Finally, we discuss some potential optimizations and ways in which the system might be tuned.

**Text filtering**

The first step in our system is to determine which sentences belong to which category. This is the more difficult problem to some extent, as we have no labeled data directly available. Therefore, a supervised learning approach is not feasible since the only means of evaluating the text filtering system is by examining its effect on the final ratings classification system. The approach we take is to filter sentences based on simple keywords, where a sentence is included if and only if it includes at least one of the specified keywords. We begin with a set of hand-picked keywords indicative of each category and then expand the keyword set by looking for words that co-occur with the hand-specified keywords.

**Hand-picked seeds**

Our first keywords were hand-picked. We designed, for each category, a set of regular expressions that matched relevant words. For instance, our patterns for dining included the following:

```java
new Pattern[] {Pattern.compile("d\*n\*"), Pattern.compile("food.\*"),
                Pattern.compile("c\*af\*es?\*"), Pattern.compile("break\*fast.\*"), Pattern.compile("l\*u\*n\*c\*h.\*")};
```

The full list of keywords can be found in ReviewFilter.java, the object that does the text filtering.

We then classified each rating with the Naive Bayes classifier described below, using these hand-picked seeds for keywords. We chose the keywords by trial-and-error, keeping those that made classification results better, and stopping once performance was reasonable and there were no obvious keywords remaining.

**Expanding the keyword set**

Our hand-picked keywords provided us with a first version of filtered text, but are fragile and require manual work to expand further. By extracting characteristics from that text, we can find new keywords and thus improve our system's recall.

We used tf-idf to find the most relevant words for each category. We considered as a document a grouping of all filtered sentences for a category, and therefore had a corpus of 11 documents (\(|D| = \{|d|\} = 11\) for each word in each category, we then compute a tf-idf score as follows:

\[
tfIdf_{i,j} = \frac{n_{i,j} \cdot \log \frac{|D|}{\sum_{k \neq i} n_{k,j}}}{|\{d : t_i \in d\}|}
\]

This should give us, for each category, the words that frequently appear in sentences about that category and rarely appear in sentences about other categories. We can then tune filters to favor precision or recall based on how many extra keywords we add.
Rating Classification

Once we have determined a way to filter the input sentences for each category, we need to train a classifier over those sentences. This is just a traditional machine learning algorithm: our data provides us with labels for each category, and we merely have to learn the labels based on the filtered text.

Bag-of-words model

To start with, we used a classic Naive Bayes classifier with unigram counts as features. We used delta smoothing with an UNK token to smooth the word probabilities, with \( \delta = 10^{-5} \). We used the tokenizer from openNLP\(^3\) to divide the reviews into words. We binarized the problem by dividing the data into "good" reviews for which the rating was 4 or 5, and "bad" ones, for which the rating was 1 or 2. Reviews with a rating of 3 or when no rating was given were ignored. In order to be able to compare the different ratings, we used training sets of the same size for each category, and with equal number of good and bad reviews.

In addition to Naive Bayes, we also performed some experiments on support vector machines. The results of all of these experiments are reported below.

Analyzing and selecting features using mutual information

Mutual information quantifies how much two random variables are correlated. We used it to find words that were correlated with the class ratings. The mutual information for two random variables is defined as

\[
I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left( \frac{p(x, y)}{p_1(x) p_2(y)} \right),
\]

where \( X \) and \( Y \) are two random variables.

For a given document (review) and a given term, we call \( U \) a random variable that takes values \( e_U = 1 \) when the document contains the term \( t \), and \( e_U = 0 \) when the document does not contain \( t \). We call \( C \) a random variable that takes values \( e_C = 1 \) when the document is in class \( c \), and \( e_C = 0 \) when the document is not in class \( c \). The mutual information between these two variables is given by

\[
I(U, C) = \sum_{e_U \in \{1,0\}} \sum_{e_C \in \{1,0\}} P(U = e_U, C = e_C) \log_2 \left( \frac{P(U = e_U, C = e_C)}{P(U = e_U)P(C = e_C)} \right)
\]

If we use the MLE probabilities computed on the training set, we get

\[
I(U, C) = \frac{N_{11} \log_2 \frac{NN_{11}}{N_{1z}N_{z1}} + N_{01} \log_2 \frac{NN_{01}}{N_{0z}N_{z1}} + N_{10} \log_2 \frac{NN_{10}}{N_{1z}N_{z0}} + N_{00} \log_2 \frac{NN_{00}}{N_{0z}N_{z0}}}{N}
\]

where \( N \) is the total number of documents, \( N_{a,b} \) is the number of documents for which the presence of term \( t \) is \( a \) (0 or 1) and the presence of class \( c \) is \( b \) (0 or 1), \( N_{az} = N_{a1} + N_{a0} \), and \( N_{0z} = N_{01} + N_{00} \).

In our case, a document is the set of all the filtered sentences relevant to a given rating. We compute mutual information for all terms and retrieve the terms that have most

\[^{3}.\ http://opennlp.sourceforge.net/
information. These are the terms that are most correlated to a given rating: the words that occur often when a rating is good and rarely when a rating is bad. This information is different from tf-idf in that tf-idf gives the words that are specific to a given rating category, such as that "waiter" that is often contained in "dining" category. In contrast, mutual information gives the words that are relevant for classifying a given category, like "awesome" or "tasty". "tasty" could be picked up by both measures, but awesome would only be picked up by MI.

In our system, we use mutual information for two purposes. First, we can use it simply to analyze the results of our classifiers and confirm that they are working as expected. Second, we can use mutual information as a feature selection method, dropping all but the most informative keywords for some threshold. The hope is that the uninformative keywords are just noise, and removing them will improve accuracy.

**Experimental Results and Discussion**

Below we describe the results of various experiments on each part of the system. In general, the text filtering component can only be analyzed indirectly (as no labeled data is available), and we are unable to precisely quantify the results of the ratings classification.

**Text Filtering**

The lack of labeled data means we have no direct means of evaluating our text filtering approach, but we can make several interesting qualitative observations.

**Hand-picked seeds**

We segmented the text into sentences, and extracted the sentences that contained specific keywords as relevant.

Some categories had obvious keywords, like dining (see above), because the reviews discussed it extensively and in a direct manner, whereas other categories needed more keywords, or less obvious ones, like entertainment:

```java
Pattern[] {Pattern.compile("entertain.*"),Pattern.compile("music.*"),
            Pattern.compile("comed.*"),Pattern.compile("tribute"),Pattern.compile("activit.*")};
```

We found the keywords like "tribute" by reading through the data, where we found that many shows on board were tributes to famous artists.

Not all keywords gave the same level of improvement on the classification performances: high frequency words, when used as keywords (e.g. "staff") had a stronger impact on performance, whether good or bad.

The final set of seeds is given below:

<table>
<thead>
<tr>
<th>Category</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dining</td>
<td>din.<em>, food.</em>, cafes?, breakfast.<em>, lunch.</em></td>
</tr>
<tr>
<td>Cabins</td>
<td>cabins?, rooms?, bed.<em>, bath.</em>, suite?, small.*</td>
</tr>
<tr>
<td>Entertainment</td>
<td>entertain.<em>, music.</em>, comed.<em>, tribute, activit.</em></td>
</tr>
<tr>
<td>Spa</td>
<td>spas?, fitness.<em>, exercis.</em>, product</td>
</tr>
</tbody>
</table>
We chose words that had positive impact of classification performance, and stopped when we reached the same classification performance as non-filtered text, in order to avoid overfitting on the test data.

**Surrounding sentences**

With our simple keyword method, in which we collect sentences that contain one or more keywords, we find obvious sentences (usually the first sentence that indicates what the subject of the paragraph is), but miss the following sentences because they talk about details of the subject and don't mention the keywords:

*Example from cabins:* The cabins are very large with lots of drawers and closet space. Not having a balcony after having an extend one on the Legend took some getting use to. **Our cabin had a problem with the wiring so constantly bulbs would burn out and were promptly replaced.** We also missed having a refrigerator -- there was more than enough space for one.

*Example from service:* **Stateroom Steward: I took the advice of another reviewer and paid the cabin steward almost right away in order to facilitate excellent service.** He was kind and replied "That really isn't necessary". Still, each morning when I shook his hand, I indiscreetly would place $10 in his palm.

Paradoxically, the more a reviewer describes a subject in detail, the more percentage of relevant sentences we miss! This impacts the classification since detailed description is full of words that would be useful for sentiment analysis.

This problem is difficult to solve with a keyword approach: we would have to include all vaguely related words and would end up selecting all the sentences. We also tried a systematic inclusion of sentences following a selected sentence, without any improvement.

Our all-or-nothing selection is too primitive, a more flexible one would be better (using weights for the key words for instance, to model the probability of a sentence being useful.)

**Non-relevant sentences**

We also pick up several sentences that contain keywords by chance, without being related to the subject:

*Example from dining:* The Dream is big - 15 decks and 3,600 passengers can leave things a bit frustrating when everyone is in a hurry (like during breakfast in port) or trying to catch an elevator from the bottom
to the top when everyone in the middle doesn't want to walk a flight or two up.

This could also be solved by weighting keywords: "breakfast" would not have enough weight in a sentence like the one above.

**Percentage of selected sentences**

By plotting histograms of the percentage of sentences selected in reviews, we see that the shape is roughly correlated with the performance. If the selected sentences are around 8-12% of the total reviews sentences, the classification works well, whereas if there is a peak towards 0, the results are bad.

![Histograms of selected sentences](image)

The three best classified categories

The three worst classified categories

Obviously, categories like "spa" that keep only a few sentences don't have enough information to base a decision on.

This is a general observation: there are also categories for which the classification is mediocre even though the percentage of selected sentences is reasonable. For the "family" rating for instance, the problem is not the percentage but the relevance of selected sentences (only the people with small children tend to comment on the family-related things, most customers don't, so finding family-related sentences is difficult.).

The bar graphs don't tell us the sentences' relevance, but they can give a diagnostic of a possible misfiltering.

In the histogram below, for the "family" rating, we see that expanding the seeds significantly reduces the number of reviews from which zero sentences are extracted, which improved accuracy:
Comparison of bar graphs of percentage of sentences selected, for the family category, with only hand-picked seeds and with 100 additional tf-idf seeds

Cognate subjects

Using the tf-idf words as seeds tends to widen the range of sentences' subjects. The "spa" category, for instance, selects sentences about massages and fitness that are cognate subjects. This is obvious in the keywords like "cardio", "toxins", "elliptical", "treadmills", "haircut", "manicure", and "pilates" (see below for some of the extra keywords extracted from each category). As an example, the following sentences were extracted from a review that gets no hits for our hand-picked keywords:

Well, goodbye expensive haircut. I was there mostly for the treadmills; DH had to use the stationery bike. The gym has relatively low ceilings and at the height of 6'2, DH was not be able to use the treadmills or elliptical without smacking his head on the ceiling. Purchased in 2006, you have been with us on many adventures - Beijing, Shanghai, Hong Kong, Thailand, Cuba, Mexico and four cruises to the Caribbean.

<table>
<thead>
<tr>
<th>Category</th>
<th>Extra words extracted by tf-idf for each category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dining</td>
<td>dinning omelet deli benedict assigned monet eggs renoir tiffany truffles</td>
</tr>
<tr>
<td>Public Rooms</td>
<td>dinning verandah cleaned whirlpool assigned monet riviera deli baggage</td>
</tr>
<tr>
<td>Cabins</td>
<td>dinning cleaned suitcases hairdryer verandah comforter bunk bedding</td>
</tr>
<tr>
<td>Entertainment</td>
<td>comedian tribute entertainer musicians jugglers dancers beatles singers</td>
</tr>
<tr>
<td>Spa</td>
<td>product elliptical treadmills exercising recreation cardio saunas exercised</td>
</tr>
<tr>
<td>Family</td>
<td>teenagers ages counselors spring traveled unruly childrens teenage</td>
</tr>
<tr>
<td>Shore</td>
<td>boats thomas canaveral drove everglades driver sand shoretrips</td>
</tr>
<tr>
<td>Embarkation</td>
<td>smoothly smooth disembark taxis boards driver breeze embark</td>
</tr>
<tr>
<td>Service</td>
<td>server attentive team directed direct headwaiter orders indonesia hello</td>
</tr>
<tr>
<td>Value</td>
<td>valuables rented driver ripped roulette 200 cent park fare priceline rent</td>
</tr>
</tbody>
</table>

The "value" category also shows improvement from cognate subjects. The classification (see following paragraph for details) improves from 59.1% to 67.2% (an 8.1% improvement). The new found keywords are decidedly money-related, with words about casinos or transport:
As a consequence, the extracted sentences are more varied, and can seem unrelated, as in this example from the "value" category (new sentences in bold):

Port Parking because the garage across the ship was full after waiting in a car line for a half hour to get in. Schlepping luggage on a tram to park. My husband was ready to take a cab to the garage before he saw a bus and went over to find out if it was "our bus. Finally gave up and left w/ my shirt and donation given. Our first tour was cancelled due to weather (no one's fault) and the second one was a "rip. They didn't get a rebate on our purchases.

A careful reading gives the impression that the sentences picked up are about people complaining in general! Since the "value" category was the one whose classification improved most with the addition if tf-idf words, we can only conclude that general complaining helps to get the general idea about good value...

**Sentence weighting**

To correct these shortcomings, we tried different variations on the sentence selection method.

In a first experiment, we tried weighting sentences rather than completely filtering out sentences that didn't match one of the keywords. In our approach, a sentence was given a weight of 1 if it did not match any filters and a weight of $w$ if it matched at least one filter. This was accomplished by duplicating each word in the sentence $w$ times, which works since we are using a multinomial Naive Bayes model. We experimented with $w=3$ and $w=100$, with the following results:

![Accuracy vs. Number of Keywords (Weighted NB)](image)

![Accuracy of Weighted NB Classifiers](image)

**Results of weighted Naive Bayes classification**

Using weighting rather than filtering, we now see a slight improvement over the baseline. In the majority of cases, using either value for $w$ performs better than the unweighted results. We see an average accuracy of 64.5% in the unweighted data, 65.0% when $w=3$, and 65.1% when $w=100$. In the extreme case of $w=100$, we're essentially performing a for of fallback: if any sentences matching the keywords are available, data from those sentences will overwhelm any data from the other sentences.
However, if no matching sentences are available, we still have the more general sentences to fall back on rather than guessing randomly. The fact that we can now make informed guesses when no matching data are available likely accounts for this model's significantly improved performance over the strict filtering model.

The second weighting strategy was trying to bypass the problems caused by the all-or-nothing sentence selection strategy. Many sentences did not contain keywords (88%), so their information was effectively lost:

![Number of keyword hits in sentences](image)

On top of this, the number of keywords present in a sentence should be an indicator of the degree of relevance, and the weight of a sentence should vary accordingly. We tried duplicating sentences proportionally to the number of contained keywords, and using a large number of keywords (200 to 2000) to get a continuous gradation of weight for as many sentences as possible.

The results were disappointing. Even if the classification performance did increase with the number of keywords, the general performance was lower than the baseline. A possible explanation is that the keywords themselves may need weighting: in the list of 2000 tf-idf keywords, the first will be very relevant but the last will carry almost no selective information. We could have given more weight to the top keywords, added the weights to obtain a sentence weight, and duplicate each sentence proportionally to its weight.

### Rating Classification

**Baseline performance**

We ran several experiments in order to precisely measure the effectiveness of both our text filtering and classification algorithms. For each experiment, we trained one classifier per category over 1,000 positive and 1,000 negative instances of that category (where a positive instance was one with a score of 4 or 5 and a negative instance was one with a score of 1 or 2). We experimented with adding varying numbers of additional keywords (0, 5, 10, 25, 50, 100, and 200), where the keywords were obtained by running the TF-IDF-based algorithm on a reduced dataset. For each category in each experiment, we also computed a baseline accuracy by training a classifier over the entire unfiltered data set.

<table>
<thead>
<tr>
<th>Category</th>
<th>Dining</th>
<th>Public</th>
<th>Cabins</th>
<th>Enter.</th>
<th>Spa</th>
<th>Family</th>
<th>Shore</th>
<th>Embark</th>
<th>Service</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>.6024</td>
<td>.6706</td>
<td>.6449</td>
<td>.5487</td>
<td>.5508</td>
<td>.4783</td>
<td>.5484</td>
<td>.6433</td>
<td>.7329</td>
<td>.7257</td>
</tr>
</tbody>
</table>

*Binary classification accuracy, with Naive Bayes classifier, without text filtering*
Stemming

We experimented a little with stemming the input data using a Porter stemmer\(^5\), but this was found to usually hurt results.

<table>
<thead>
<tr>
<th></th>
<th>Dining</th>
<th>Public</th>
<th>Cabins</th>
<th>Enter.</th>
<th>Spa</th>
<th>Family</th>
<th>Shore</th>
<th>Embark</th>
<th>Service</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unstemmed</td>
<td>.6024</td>
<td>.6706</td>
<td>.6449</td>
<td>.5487</td>
<td>.5508</td>
<td>.4783</td>
<td>.5484</td>
<td>.6433</td>
<td>.7329</td>
<td>.7257</td>
</tr>
<tr>
<td>Stemmed</td>
<td>.6024</td>
<td>.6659</td>
<td>.6495</td>
<td>.5394</td>
<td>.5508</td>
<td>.4976</td>
<td>.5368</td>
<td>.6421</td>
<td>.7051</td>
<td>.7048</td>
</tr>
</tbody>
</table>

*Accuracy for one experiment comparing the effect of stemming data*

Even using the same training set, we get significantly different accuracies in each category. Clearly, some ratings are harder than others. The performance on "service" and "value" is good, but "family" gets an accuracy lower than random guessing in this experiment.

Adding tf-idf keywords

The first experiment analyzed the simple Naive Bayes algorithm using different numbers of generated keywords. The goal was to see whether the filtering-based approach could provide better results than a baseline classifier trained over all of the input data, and to find the ideal number of keywords to use. The results are shown below:

![Results of basic Naive Bayes classification](image)

In general, this form of filtering did not improve accuracy. Compared to the baseline unfiltered results, we see slight improvements in two categories (Entertainment and Family), whereas all other categories are harmed somewhat. The average accuracy is 64.5% for the unfiltered data and 61.6% for the filtered data. The ideal number of seeds is inconsistent, with additional seeds increasing accuracy in some places and decreasing it in others. Clearly we needed to take a slightly more intelligent approach.

SVM classification

We experimented with using a classifier based on support vector machines rather than Naive Bayes. SVMs are generally considered to be more powerful classifiers than Naive Bayes, as they do not necessarily assume all data to be independent. For this

---

experiment, we used the LIBSVM\(^6\) package rather than implementing SVMs ourself, and filtered sentences strictly rather than weighting them. The results are shown below:

![Accuracy vs. Number of Keywords per Category (SVM)](image)

**Results of support vector machine classification**

The first thing to notice is that general accuracy is significantly higher than for Naive Bayes, with the support vector machine predicting the correct overall rating 80.4% of the time vs. the Naive Bayes classifier's 68.3%. This is consistent with the general view of SVMs as the more powerful machine learning algorithm. However, the performance gap between the unfiltered and filtered reviews is significantly larger, with review filtering harming accuracy to a significant extent. This also makes some sense --- SVMs are more powerful than Naive Bayes and not as beholden to the independence assumption, so they are better able to do their own work of deciding which data are the most relevant. By filtering out data, we are merely limiting the information at the SVM's disposal.

**Using mutual information keywords as features**

Finally, we can experiment with the results of analyzing and filtering the keywords based on mutual information. The top few keywords for each category are shown below. They vary significantly from the tf-idf keywords: whereas the tf-idf keywords denote words that separate one category from another, the mutual information keywords separate positive from negative instances within that category. These words are fairly similar in each category ("great" is one of the best words in nearly every category), but there is also some interesting variation between categories. For example, rooms are often described as "beautiful" or "dirty", whereas spa and shore excursion are more likely to be described as "fun".

<table>
<thead>
<tr>
<th>Category</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>great, 2003, loved, lunch, enjoyed, beach, tour, told, fun, ate, worth</td>
</tr>
<tr>
<td>Dining</td>
<td>enjoyed, deck, loved, ride, walked, ate, terrible, excellent, early</td>
</tr>
<tr>
<td>Public Rooms</td>
<td>beautiful, dirty, airport, nice, early, shows, luggage, flight, good, quickly</td>
</tr>
<tr>
<td>Cabins</td>
<td>great, beautiful, tour, shopping, lunch, nice, early, luggage, breakfast</td>
</tr>
<tr>
<td>Entertainment</td>
<td>great, lunch, deck, taxi, fun, went, arrived, luggage, terminal, both</td>
</tr>
<tr>
<td>Spa</td>
<td>fun, lunch, poor, nice, island, worth, ;, took, friendly, cayman, deck</td>
</tr>
<tr>
<td>Family</td>
<td>loved, great, drove, plenty, tour, nothing, wall, not, favorite, beach</td>
</tr>
<tr>
<td>Shore</td>
<td>great, fun, nice, shopping, ;, ate, well, plenty, went, night, deck</td>
</tr>
<tr>
<td>Embarkation</td>
<td>wonderful, shopping, went, beautiful, morning, good, shops, much, st</td>
</tr>
</tbody>
</table>

Although mutual information is a useful way to examine what exactly the classifier is doing, it is less useful as a way to directly filter results. We see below that accuracy tends to increase as we increase the number of available features, with a slight peak at 600 features that may just be attributable to noise. As Naive Bayes classifiers perform their own weighting of features, presumably the spurious features are weighted low enough that removing them makes little difference.

![Number of Features vs. Accuracy](chart.png)

*Overall accuracy as a function of the number of mutual information features retained*

**Future Work**

Many possible improvements over our system remain. We worked primarily with Naive Bayes classifiers, but our few experiments with support vector machines suggest that they could lead to a much higher classification accuracy if optimized correctly. Since the SVM extracts meaningful features by itself, we could try using that information to filter relevant sentences.

There are also many different ways to optimize the current filtering algorithms using different weighting or keyword-finding strategies. However, because the quantity of text describing different categories varies greatly across reviews, reviewers and categories, we don't expect to get the same performance for each category: family and spa will always be difficult to classify for example.

Of course, it would also be useful to obtain additional data. Since most reviews are positive (roughly 90%), we are limited primarily by the number of negative reviews available. Even more useful than gathering additional ratings, however, would be to acquire labeled data specifying the category of each sentence. This would allow us to tune the sentence classifier directly rather than measuring its indirect impact on the entire system.

**Conclusion**

Our system consisted of two major components: classifying sentences as belonging to certain rating categories, and classifying reviews into positive and negative ratings for each category. We are fairly pleased with the performance of the first part of our system. Though the lack of labeled data means we cannot directly measure our
accuracy, it seems to perform well anecdotally and the keywords extracted by tf-idf are reasonable. Ours is a flexible approach that could easily serve as a basis for more sophisticated algorithms.

Classification of the ratings themselves did not perform as well as we had hoped. Simply training a classifier for each category and classifying using the entire review works fairly well and is difficult to improve upon. Strictly filtering out sentences that seem irrelevant only hurts performance as it makes too little information available to the classifier in many circumstances. However, weighting relevant sentences rather than filtering irrelevant ones does improve accuracy a small amount (generally less than 1%).