Extracting Common Sentiments From Reviews

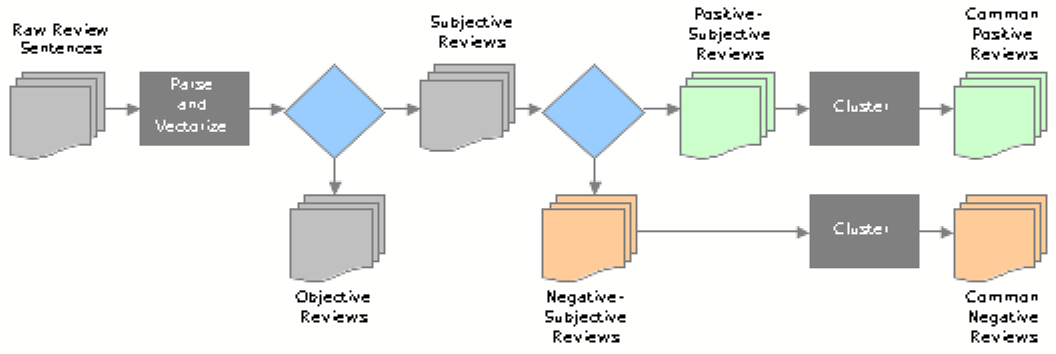
A large and growing body of user-generated reviews is available on the Internet, from product reviews at sites like Amazon.com to restaurant reviews at sites like Yelp.com. For users making a purchasing or dining decision, the opinions of others can be an important factor. Although some aggregate information -- like average star ratings -- for multiple reviews is sometimes available, in general the only way to get a sense of the overall sentiment among users is by reading through many reviews. As the number of reviews for a single product or restaurant becomes large (on the order of hundreds or even thousands), it becomes increasingly impractical to read every review.

We view the goal of reading multiple reviews as finding widely-held opinions and weighing the positive against the negative, and we wish to automate this sort of task using NLP and machine-learning techniques. The problem can be broken down into three major components: sentence-level sentiment classification; sentiment clustering and ranking; and summarization. Sentiment-classification involves labeling every sentence in every review for a particular restaurant as either *Subjective-Positive*, *Subjective-Negative*, or *Objective*. A range of literature exists on this problem. Pang et al. describe the successful use of traditional machine-learning techniques, such as Naive-Bayes, for the sentiment classification of entire movie reviews. [1] Hu and Liu propose a system to solve a very similar problem to the one we pose. [2] Instead of simple classification, they approach the problem by first extracting opinion words from each sentence and then predicting the polarity of the sentence by the dominant polarity of its constituents. They grow sets of positive and negative opinion words using seed words in WordNet. Given the success of Pang et al. with simple classification techniques, we plan to take this approach, exploring various feature sets and classifiers. After isolating subjective sentences from objective sentences, we will cluster those subjective sentences that are closely-related using a simple K-means algorithm and rank the resulting clusters using a cluster-quality metric that rewards large, cohesive clusters.

In the remainder of this paper we will present our proposed system, justify various design decisions, and discuss the performance of the final system.

Proposed System

The figure below illustrates the multiple stages of processing that result in a ranked-list of closely-related opinions expressed in a set of reviews.



Data

We use two sets of data in this project. First, for training our subjectivity and polarity classifiers, we take advantage of publicly available movie review data [3]. Specifically we use the "Subjectivity dataset V 1.0" [4] and the "Sentence Polarity dataset V 1.0" [5]. The subjectivity dataset consists of 5,000 objective and 5,000 subjective sentences. The subjective sentences come from Rotten Tomatoes [6] reviews, while the objective sentences come from IMDB [7] plot summary snippets. The training data are not perfect: Plot summaries may contain subjective opinions, and sometimes reviews contain objective information (though the latter is less likely). The objectivty dataset consists of 5,331 positive and 5,331 negative sentences. The sentences come from Rotten Tomatoes "Fresh" and "Rotten" reviews respectively.

The data we wish to extract opinions from were collected from Yelp.com. In total there are 1731 reviews of 6 restaurants, with each review containing an average of 11 sentences.

Classifiers and Features

We experimented with both Naive Bayes and SVM classifiers, using a number of different features. The following shows how we carefully chose the best combinations through multiple design iterations with intermediate error analysis.

Generation 0: Bag of Words and a Naive Classifier

Our initial approach is to treat each sentence as a bag of words. We represent a sentence using a vector whose entries correspond to the *TF-IDF*-weighted frequency of each of the words in the vocabulary. We use *TF-IDF* weighting to get a better representation of the importance of each word. It also obviates the need for the use of stopwords, which might have been detrimental to the performance of the classifier. To keep the dimensionality of the resulting vectors manageable we limit our vocabulary to the N most-frequently-occurring words.

Our initial classifier is Naive Bayes (NB). Its performance is decent for subjectivity classification but rather poor for polarity classification. We present our initial results below.

Results| Classifier: NB. Features: Bag of words. Vocabulary size (N): Approximately 1000 per class.

Classification Task	**Train Accuracy (%)**	**Accuracy (%) [5-fold CV]**
Subjectivity	82.08	**81.65**
Polarity	63.9655	**62.9244**

Generation 1: Stems of Words and a Sophisticated Classifier

We reason that using stems instead of words should have a two-fold beneficial role to our problem. First, it will decrease the sparseness in the data, since there are fewer distinct stems compared to distinct words. Second, it should be able to capture and group better semantic information. To that end we employ the Snowball [8] algorithm and generate bags of stems instead of bags of words. We still use *TF-IDF* when constructing stem vectors.

Moving from words to stems increases subjectivity classification accuracy by 1 full percentage point (to 82.62%) and polarity classification accuracy by 1.5 percentage points (to 64.4251).

We also proceed by training more sophisticated classifiers than NB. Specifically we investigate several members of the Support Vector Machine (SVM) family with different Kernels. Our results are presented below.

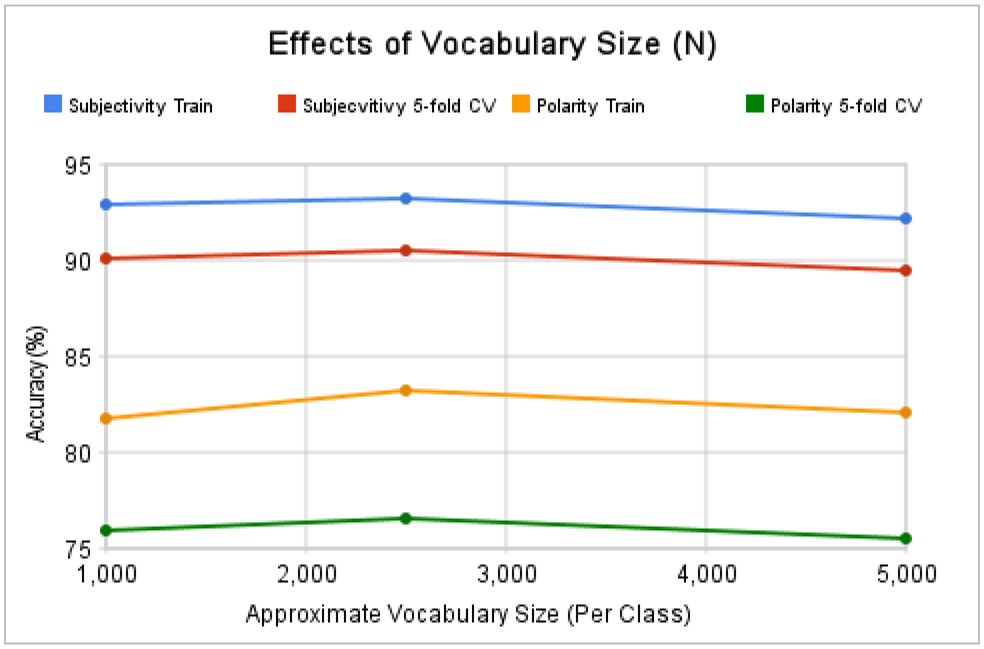
Results | Features: Bag of stems. Vocabulary size (N) : Approximately 1000 per class.

Classification Task	**Classifier**	**Kernel**	**Train Accuracy (%)**	**Accuracy (%) [5-fold CV]**
Subjectivity	NB	N/A	82.93	**82.62**
Subjectivity	SVM	Linear	97.32	**85.67**
Subjectivity	SVM	Sigmoid	91.86	**89.73**
Subjectivity	SVM	Radial Basis Function	92.89	**90.12**
Polarity	SVM	Radial Basis Function	81.767	**75.9426**

It is apparent that use of sophisticated classifiers results to a sizeable improvement in performance. Specifically, switching to an SVM with a Radial Basis Function (RBF) Kernel [exp(-gamma*|u-v|^2)] increases accuracy by approximately 8 percentage points for subjectivity to an impressive 90.12%, and by approximately 11 percentage points for polarity to an acceptable 75%.
It is notable that the Linear Kernel [u'*v] showed considerable over-fitting, though it still outperformed NB. The Sigmoid Kernel [tanh(gamma*u'*v + coef0)] performed almost at par with the RBF Kernel, but we chose the latter for the remainder of our analysis.

Generation 2: Increasing the Vocabulary Size (N)

In this generation we attempt to investigate the effect of the Vocabulary size (N) in our classifier's performance. So far we have been limiting N to approximately 1000 words per class (eg Subjective vs Objective). By doing so we decrease the dimensionality of the problem but we could be neglecting low-frequence stems that have highly valuable semantic information. We therefore experiment with different vocabulary sizes in order to determine the optimal size. Experimentation shows that N=2500 seems to be a sweet spot, with N=1000 leading to underfitting and N=5000 leading to overfitting. As a rule of thumb using N = 0.25 * NumberOfTrainingSentences gives good results. The results are clearly depicted in the following figure.



Using our best classifier from Generation 1 and using N = 2500 we get a moderate increase (clost to half a percentage point) in accuracy for both subjectivity and polarity classification. Our results at the end of Generation 2 are summarized below.

Results| Classifier: SVM-RFB. Features: Bag of stems. Vocabulary size (N): Approximately 2500 per class.

Classification Task	**Train Accuracy (%)**	**Accuracy (%) [5-fold CV]**
Subjectivity	93.22	**90.55**
Polarity	83.2114	**76.5616**

We investigate the confusion matrices for the two classification tasks.

```
=== Subjectivity ===
    a    b   <-- classified as
 4674  326 |     a = Subjective
  619 4381 |     b = Objective


=== Polarity ===
    a    b   <-- classified as
 4029 1302 |     a = Positive
 1197 4134 |     b = Negative
```

Almost twice as many objective sentences are misclassified as opposed to subjective. Misclassification rates are relatively similar between positive and negative sentences. We therefore need novel features that will add predictive power to our classifiers.

Generation 3: Integrating Parser Features

We proceed by integrating Part-Of-Speach (POS) features in the classifier by using the Stanford Parser. [9] The rational behind this is quite simple: in terms of subjectivity classification we expect rather different sentence structure; subjective sentences should in general have more adverbs and adjectives. To that end we add POS tags as features and perform the same *TF-IDF* vectorization as we did with stems. In essence then, POS tags are treated as stems themselves. The approach succesfully increases the subjectivity classifier's accuracy but only marginally so.

Syntactic structure should not differ between positive and negative sentences. However we can attempt to capture more of the semantic information by identifying stem bigrams in order to improve our polarity classifier. To avoid sparsity issues we don't generate all possible bigrams but only generate the ones that appear to be syntactically sound. Technically speaking we only generate bigrams for subtrees of nodes that are pre-pre-terminals. The approach successfully increases the objectivity classifier's accuracy by slightly more than half a percentage point.

Our results for the best classifiers for each task are summarized below.

Results| Classifier: SVM-RFB. Features: Bag of stems. Vocabulary size (N): Approximately 2500 per class.

Classification Task	**Features**	**Train Accuracy (%)**	**Accuracy (%) [5-fold CV]**
Subjectivity	Bag of stems, pos	93.24	**90.82**
Polarity	Bag of stems, syntactic-bigrams	83.5115	**77.2369**

Notably, for both the classifiers the 5-fold CV accuracy increased more than the train accuracy, increasing our confidence in the quality of the new features.

Once again we investigate the confusion matrices in search of understanding.

```
=== Subjectivity ===
     a    b   <-- classified as
  4673  327 |    a = Subjective
   591 4409 |    b = Objective
```

Indeed it seems that POS features helped in decreasing the Objective misclassification rate, but do so less than we would have expected. We reasoned that perhaps the training data are rather noisy and indeed, consider the following sentences that appear as objective training data:

- `but what exactly is good & what exactly is evil?`
- `eleven year old david wiseman is mad about cricket but no good at it.`
- `it will definitely keep you on the edge of your seat because of the blood and gore or the good looking babes.`

Each of this sentences could very well be interpreted as subjective by a human reader. The presence of adverbs and adjectives does not help that much in their classification. We reason then that our classifier has reached a sufficiently good performance and getting higher than that might border the lines of over-fitting to the training corpus.

```
=== Polarity ===
     a    b   <-- classified as
  4064 1267 |    a = Positive
  1160 4171 |    b = Negative
```

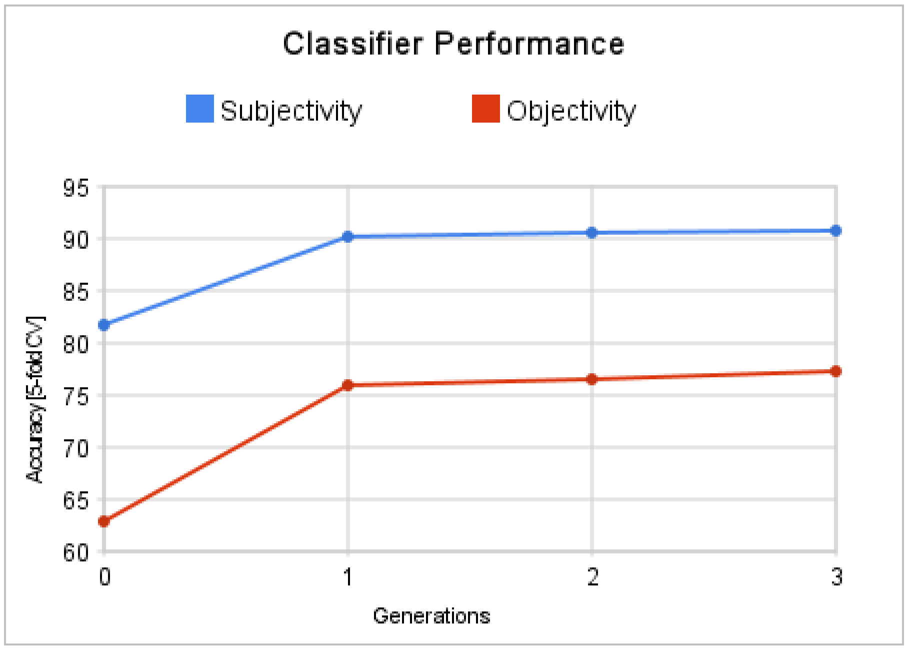
Polarity classification seems a more difficult task in general. The problem seems especially difficult if a bag-of-items (stems,pos etc) is used because people tend to mix in their reviews positive and negative opinions. Consider for example:

- `Positive: "slow but clever"`
- `Negative: "clever but slow"`

Our current approach cannot distinguish very well between the two (we would have said "cannot distinguish at all", but the syntactic bigrams might have some effect, although they suffer from data sparsity). Yet for people, the adjective after the "but" carries a greater weight. Looking at our training data reveals that this "but" motif is quite prevalent in reviews. Perhaps then a feature that would associate adjectives and adverbs with their position relative to a present "but" would increase the ability to differentiate between positive and negative. Similarly "very good" and "not very good" cary opposite semantic meanings but the bag approach is not so good at capturing that. Use of syntactic information from the parser could facilitate building of more semantic features. We propose such an approach for future work.

Generations Summary

The best increase in performance came by choosing sophisticated classifiers (SVMs). Using stems and parser features shows mild improvements in both classification tasks but more so in the case of the objectivity task. The objectivity classification task is open for future improvements.



Informative Features

To determine the significance of features we looked at the information gain of each feature with respect to the class [InfoGain(Class,Attribute) = H(Class) - H(Class | Attribute)].

With respect to subjectivity classification, some of the most important features are:

```
STEM_it , STEM_his , POS_RB , STEM_. , STEM_movi , STEM_he ,
STEM_her , STEM_film , STEM_) , STEM_( , POS_JJ , POS_NN ,
POS_. , STEM_she , POS_PR
```

Some of the features, such as `POS_*` should work well accross domains, however some of the features are very domain specific and also very much tailored to the training data: objective sentences come from plots and use lots of pronouns to refer to movie characters, places and situations. On the other hand subjective sentences come from movie reviews which often refer to the "movie" or "film" being reviewed (eg "the movie/film was not that great"). Also movie reviews tend to use parenthesis much more than plots.

With respect to polarity classification, some of the most important features are:

```
STEM_bad , STEM_and , STEM_too , STEM_bore , STEM_beauti ,
STEM_dull , STEM_perform , STEM_of , STEM_movi , STEM_refresh ,
STEM_no , STEM_heart , SYNTACTIC_BIGRAM_the_best , STEM_just ,
STEM_film
```

Not surprisingly, adjectives like "bad", "boring", "dull", "refreshing" are good indicators of the polarity classification of the movie. Similarly "too" and "just" are most often (thought not always) used in negative sentences. Some of the features are domain specific and some are not.

Cross-Domain Applicability

We had hoped that classification power would extend to domains beyond the narrow area of movie reviews but analysis of informative features suggests otherwise. We go ahead and evaluate our classifiers on some hand-labeled subjectivity Yelp data (838 objective and 426 subjective sentences). The results are depressing. We get an accuracy of 43.038%. The confusion matrix is presented below.

```
=== Confusion Matrix ===
   a   b   <-- classified as
 374  52 |   a = Subjective
 668 170 |   b = Objective
```

It is evident that the classifier performs very badly (worse than random guessing) when applied across domains. The majority of the error comes from misclassification of sentences as subjective. We investigate significant features in the hand-labeled Yelp subjectivity data by once more using information gain. Some of the top features are:

```
STEM_was , STEM_great , STEM_excel , STEM_delici , STEM_good ,
STEM_veri , STEM_atmospher , STEM_is , STEM_servic , STEM_amaz ,
STEM_the , STEM_food , STEM_realli , STEM_got , STEM_call
```

Not suprisingly, some of the most important features tend to be domain specific (e.g. atmosphere, service, food, delicious). We expected performance to drop when moving across domains, but the nearly worse-than-chance performance renders the classifier unusable.

In an attempt to remove semantic biases we build a classifier that relies solely on `POS_*` features. We expect it to be unbiased with respect to domains. The results are presented here.

Results| Classifier: SVM-RFB. Features: Bag of POS.

Classification Task	**Train Accuracy (%)**	**Accuracy (%) [5-fold CV]**	**Test Accuracy (%) [On Yelp Data]**
Subjectivity	93.22	**90.55**	**50.9494**

And the corresponding confusion matrix on the test set is:

```
=== Confusion Matrix ===
    a   b   <-- classified as
  329  97 |   a = Subjective
  523 315 |   b = Objective
```

The classifier performs better than chance but only marginally so. The main problem still seems to be misclassification of sentences as objective.

We hypothesize that the sentence form differs across review domains not only in a semantic way but also in a syntactic one. To validate our hypothesis we train subjectivity classifiers on the Yelp data and report the results.

Results| Subjectivity Classifier: SVM-RFB. Vocabulary size (N): Approximately 250 per class.

Features	**Train Accuracy (%)**	**Accuracy (%) [5-fold CV]**
pos	69.7785	**66.9304**
stems,pos,syntactic-bigrams	81.4082	**72.7057**

The difference in syntactic structure between movie and Yelp data can be seen by contrasting the 50.9% with the 66% accuracy of the differently trained classifiers. Perhaps a larger training size for Yelp data would have given us better results. Perhaps better quality data would have given us better results. Case in point:

- `plouf is my unicorn, my elanor.`
- `*ploof* aaaand i'm gone!`

We conclude that training a classifier for a specific domain, even with a small amount of data is a better choice than building a cross-domain classifier. The option of training a generic classifier with training data from multiple domains has not been investigated in this paper and remains open.

Clustering

Given two subsets of review sentences from the full review set that have been classified as *Subjective-Positive* and *Subjective-Negative*, respectively, we need to group the sentences together into clusters of commonly-expressed opinions. To do so, we need to choose a clustering algorithm, extract features from each sentence, and find the number of clusters, *K*, that maximizes cluster quality.

Design Considerations

The WEKA library provides us with several clustering algorithms, from simple *K*-means to EM. As we will discuss in the cluster quality section below, in addition to clustering our application requires a method to measure the quality of individual clusters against an application-specific metric, both for choosing an optimal number of clusters and for ranking the resulting clusters. Unfortunately, it proved impractical to get this kind of information from WEKA. We attempted to use a feature of the EM clusterer that optimized the number of clusters, *K*, by maximizing the probability of generating them, but it tended to pick *K* much too small, resulting in extremely large clusters of mostly unrelated opinions.

Instead, we chose to implement a custom *K*-means clustering algorithm in MATLAB that allowed us to evaluate individual cluster quality with a custom metric. Our goal for clustering has been stated as finding widespread, common opinions. Two cluster properties help formalize this notion. A cluster is more likely to contain sentences about the same thing if the distance between sentences is small, and we can measure this with the residual sum of squares (RSS). Further, the more widespread an opinion is, the larger its cluster will be. Individually, neither measure helps us determine the quality of a cluster, because RSS goes to zero as K gets large and average cluster size increases as K goes to zero. But we can combine the two measures into a cluster quality measure, Q, which simultaneously penalizes clusters that are large but diffuse or dense but small. The quality of a particular cluster, *j*, is given by:

$$Q(j) = \frac{1}{\alpha(1 + size_j) + \beta\frac{1}{1+RSS_j}}$$

The values of *α* and *β* affect the relative contribution of each component to the quality measure, and their choice is a design consideration. One simple way to choose them, and the method we used, was to run clustering with a guessed value of *K*, find qualitatively good clusters in the results, and then choose coefficient values that reward them. Using this method, we chose *α*=0.01 and *β=1.* We use this quality measure for two purposes. First, we can choose an optimal value of *K* by maximizing average cluster quality over a reasonable search space. Second, given an optimal *K*, we rank each cluster by its internal quality measure so that the best clusters are likely to be first in an ordered list, which we can then use for summarization.

The feature set for clustering is fundamentally different than for subjectivity and polarity classification. For example, to distinguish a subjective sentence from an objective sentence, we are looking for features that correlate with sentiment, like adjectives, but we don't care what noun the adjective is modifying. In fact, we would ideally like to ignore the noun being modified, as it could potentially diminish classifier performance. However, when clustering we already have a set of sentences classified as subjective, and we would like to find *what* a subjective sentence is expressing an opinion about. So we care more about the noun in this case. Also when clustering, we need to pay attention to word frequency, since high-frequency words can easily lead to undesirable clusters of sentences that all contain, for instance, the word *the*.

Feature Analysis

In order to determine what features result in the best clusters, we experimented with a number of combinations, including individual words, stemmed words, and stemmed words with syntactic bigrams. The parser and feature extractor from classification were reused. During vectorization, a stop list was used to eliminate high-frequency words like *the* and *I*, and the *ith* element of a sentence vector contained the count of feature *i*.

The simplest feature set, individual words, was quite effective and produced many good clusters. For example, here's a portion of a high-ranking cluster from the restaurant Evvia, found with word features:

- `we had the lamb chops again but this time it wasn't served as hot`
- `evvia has hands down the best lamb i've ever ordered`
- `but overall the lamb that was on the prefix menu was more than decent`
- `the lamb shank was to die for`
- `friends had lamb and chicken`
- `for our entrees we had the lamb chops and the goat stew`
- `lamb chops were great as expected`

Clearly, the strong feature shared by all sentences is the word *lamb*. Note that because bi-grams are not used as features, there is no distinction between the *lamb chops* dish and the *lamb shank* dish. Also, some of the sentences are not opinion, which is an artifact of the poor classifier performance. Unfortunately, using word features can just as easily form clusters of sentences that are similar but don't express common opinion:

- `this is a great place for a romantic date`
- `holy cow this place is amazing`
- `this place is loud`
- `overall, this place is just ok for me`

All of the opinions are talking about *this place*, but the opinions vary widely, in part due to poor classifier performance. But beyond this, it's clear that the words that define the cluster simply have little to do with a specific opinion.

In order to reduce sensitivity to word form, we also experimented with stemming, which generally improved clustering results. Here's an example of a cluster whose formation was aided by stemming the words *pricey, price, and prices*:

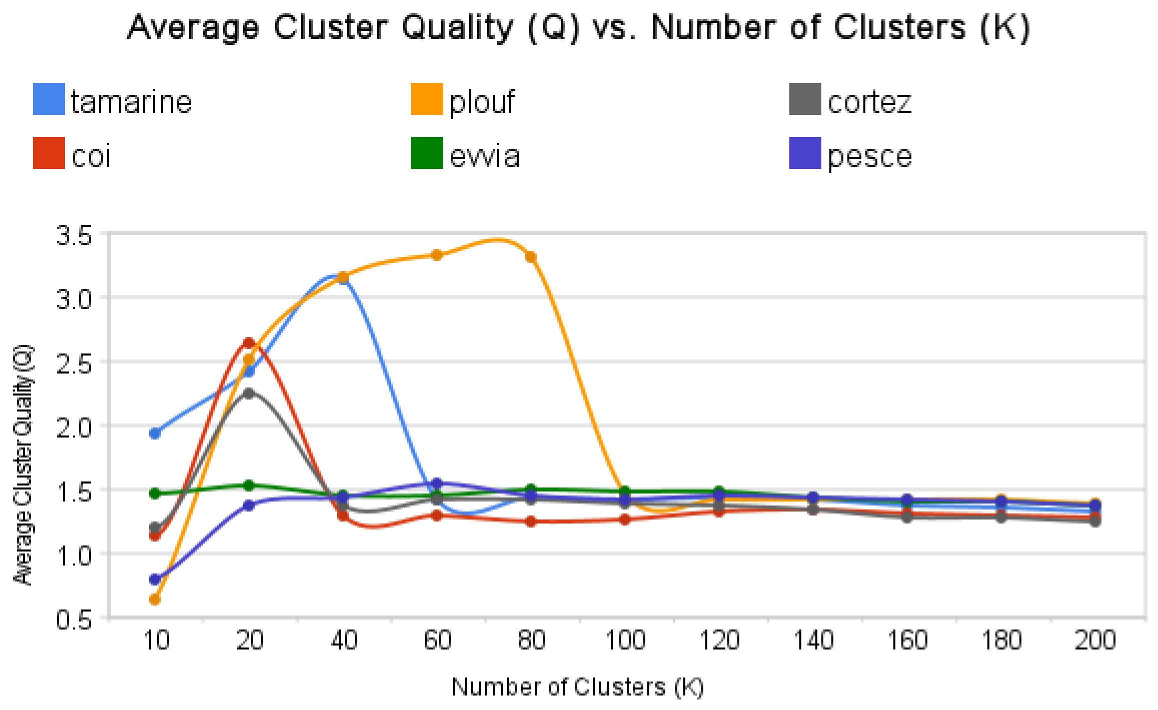
- `pricey, but well worth it.`
- `definitely on the pricey side but the food, service and experience is truly worth it.`

- `the prices were a little high, but it was worth it and sometimes you just have to do it.`
- `totally worth the $23 price tag.`
- `it's a little pricey ($15 for a lamb gyro) but worth it`

Adding syntactic POS tags to the feature set substantially reduced cluster quality both quantitatively and qualitatively, as expected. This is likely because it's equivalent to adding high-frequency words to the sentence vectors, making it easier for unrelated opinions to become related on the basis of their part-of-speech composition. Adding syntactic bi-grams had little impact on performance, probably because the clustering algorithm was already able to group sentences in which certain words frequently occurred together, like *Palo Alto*.

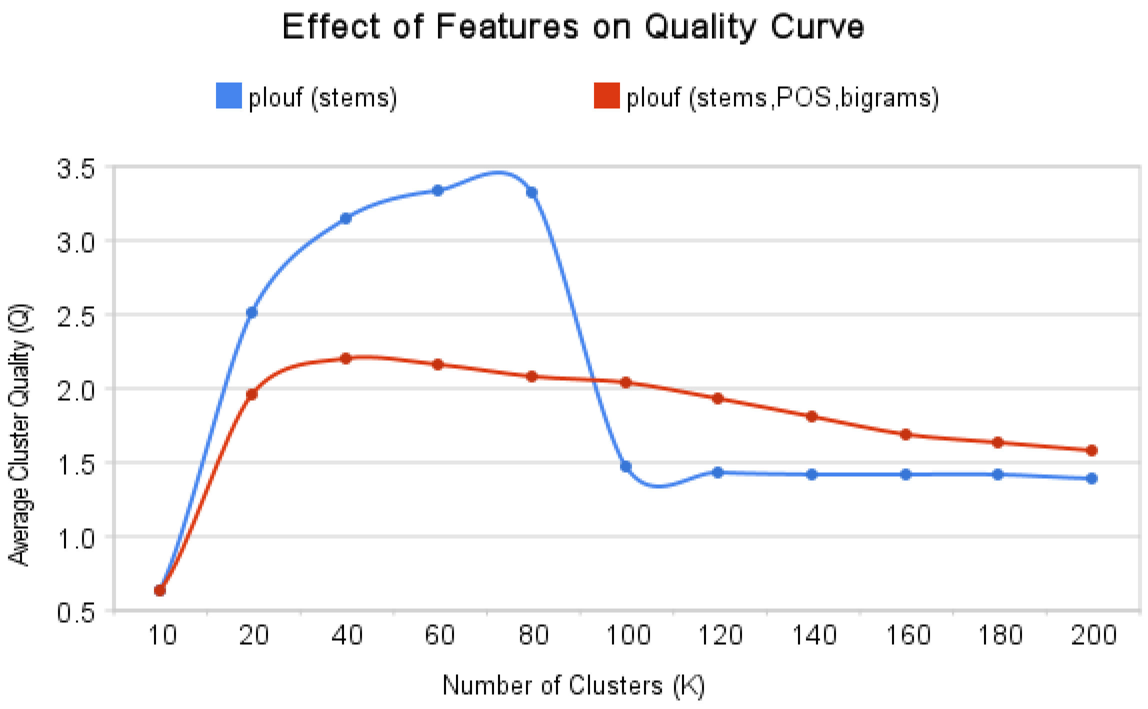
Cluster Quality Optimization

Because we don't know *a priori* how many clusters exist in the data, we need a method for finding the optimal value. We experimented with choosing *K* on the basis of average cluster quality, using the quality metric previously discussed. The graph below shows how the quality varies as a function of *K*, with vectors of stemmed-word counts.



For the reviews of most restaurants, there is a clear peak in quality, suggesting that the clustering algorithm is actually finding structure in the data. As expected, quality tends to start out low, with only a few, low-quality clusters, and as *K* gets large and approaches the total number of sentences, cluster quality correspondingly goes down as good clusters begin to fragment.

We've previously observed that adding POS tags to the feature set hurts performance, leading to less-meaningful clusters. Comparing quality curves helps to make this notion more formal. In the figure below, we see that adding POS and bigram features substantially reduces the peak quality for clusters from the restaurant Plouf. This suggests that the features, while useful for classification purposes, in fact mask the inherent structure in the data that leads to good clusters.



Final Results And Future Work

Overall, we have shown that SVM classifiers perform well when classifying the subjectivity of sentences within the same domain. They are also effective, though less so, at classifying the polarity of subjective sentences in the same domain. Unfortunately, our key hypothesis -- that we could train classifiers with movie-review data and transfer them to a different domain -- proved incorrect. At 51% accuracy on Yelp test data, the subjectivity classifier was completely unusable. The high accuracy we acheived with the classifier on movie-data seems to have come at the cost of over-fitting to the data of a single domain. Looking at the features with highest information gain gives some insight into why this is the case. Stems like "movi" and "film" are unlikely to be informative in a restaurant-review domain.

One potential solution to this poor cross-domain performance is to do more work upfront to extract the particular types of features that are informative of opinion. Something like the approach of Hu and Liu -- extracting opinion words -- would seem to be appropriate. Alternatively, combining training data from multiple domains would also potentially be helpful.

Despite the shortcomings of the classifier, clustering overall worked well, though it was difficult to give any rigorous evaluation of the usefulness of the results. Indeed, developing some way of having users rate the quality of top-ranking clusters would probably be a useful next step to assessing the effectiveness of our approach. Nevertheless, we have developed a framework for extracting common opinions from a review-sentence set in an entirely unsupervised manner, with the proposed quality measure as an integral part of its design.

The most difficult part of classification seems to be the proper selection of features. Its apparent that the notion of an opinion is quite difficult to capture with simple features like words, stems, or parts-of-speech. In fact, the subtly of opinion remains one of the most challenging obstacles to true sentiment classification. The movie-review classifiers worked as well as they did mainly because there was a substantial difference between the features of movie plot summaries and movie reviews. These differences were far less significant within Yelp reviews.

One common aspect of good clusters is that they seem to form around specific aspects of a restaurant, like *atmosphere*, *lamb falling off the bone*, *service*, or *wok pho noodles*. Poorer, but still high-scoring, clusters tend to form around high-frequency words that the stop-word list did not remove. This suggests a slightly different approach to building feature-vectors for each sentence. First, a pre-processing pass could build a list of words and phrases that appear frequently in the review of a particular restaurant but are uncommon in the wider corpus. This should find phrases like the name of a dish that many people are talking about. Second, given the narrow domain of the problem, it should also be possible to hand-build a list of common ideas a reader might want to know about, like *service*, *food*, and *price.* Extracting these combined, specific features should lead to purpose-built vectors that form clusters around relevant concepts.

References

[1] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86.

[2] M. Hu and B. Liu. Mining opinion features in customer reviews. In *Proc. AAAI*, 2004.

[3] http://www.cs.cornell.edu/People/pabo/movie-review-data/

[4] Bo Pang and Lillian Lee, A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts, *Proceedings of ACL 2004*

[5] Bo Pang and Lillian Lee, Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, *Proceedings of ACL 2005*.

[6] http://www.rottentomatoes.com/

[7] http://www.imdb.com/

[8] http://snowball.tartarus.org/

[9] http://nlp.stanford.edu/software/lex-parser.shtml