Pro, Con, and Affinity Tagging of Product Reviews

Todd Sullivan
Department of Computer Science
Stanford University
todd.sullivan@cs.stanford.edu

1 Introduction

As internet use becomes more prevalent, an increasing number of consumers are turning to product reviews to guide their search for the best product. This is especially true for products that the consumer may not have extensive knowledge about, such as cameras and GPS devices. According to a 2006 eTailing Group and JC Williams Consultancy study, 70% of online shoppers examine customer reviews before they buy and 92% find customer reviews to be "very helpful" in making a decision, which makes product reviews an important facet of consumer decision making and an important aspect of retailing and e-commerce.

While plain text reviews are helpful, most consumers do not have enough time to read many reviews about multiple products. As a result, reviews are much more helpful if they contain tags for the pros and cons. Most consumers would agree that reviews are even more helpful if the product's main page that contains all of its reviews contains the aggregate information of these tags.

In this paper, we consider the task of assigning pros, cons, and affinities (more on these later) to product reviews given information such as the review's title, comment text, and rating. We begin in Section 2 with background research in extracting pros and cons from reviews and describe our problem definition in Section 3. We then present our baseline systems that use a bag of word approach with a Naïve Bayes classifier (Section 4). In Section 4 we also examine the implications of various preprocessing techniques such as removing punctuation, lowercasing words, and stemming. In Section 5 we present a maximum entropy classifier that shows considerable performance increases over the baseline system. We move to making joint decisions on tag sets in Section 6, and finish with a brief listing of the many ideas and techniques that we did not have sufficient time to incorporate into our systems.

2 Background

There has been many successful previous works pertaining to analyzing reviews. For example, Bo Pang, Lillian Lee, and others have had much success in predicting the rating, or sentiment, expressed in review text [1, 2, 3]. In their work, they found that sentiment classification was by-and-large more difficult than standard topic-based categorization, with many successful techniques in topic classification such as incorporating frequency information into n-grams actually decreasing performance in the sentiment classification setting. Additionally, they found that unigram presence features were the most effective (none of their other features provided consistent performance boosts when unigram presence features were used), and that bigrams were not effective at capturing the context.

Other works, such as that of Soo-Min Kim and Eduard Hovy [4] have explored extracting pros and cons from reviews. In their work, they found that pros and cons occur in both factual and opinion sentences within reviews. Their results hint at the possibility of using different tactics to extract tags based on whether the sentence is determined to be factual or an opinion.

3 Problem Definition

Our setting is slightly different from previous studies. We have obtained product review data for the GPS Device category of Buzzillions.com, which is a product review portal run by PowerReviews. PowerReviews offers customer review solutions to e-commerce businesses.

Their system collects reviews from consumers that are verified to have purchased the product from an e-commerce store in their network. In addition to a product rating, title, and comment text, the customer is given the option to input pros, cons, and affinities as tags. Affinities are summarizations of the type of consumer that the customer is, such as the affinities "Frequent Traveler," "Bad With Directions," and "Technically Savvy" in the GPS Device category. The user can input their own tags for these sections as well as choose from four to ten tags that are either preselected by PowerReviews moderators or are the most popular tags in the category.

Instead of extracting pro and con sentences from the comment text, we use the most frequently inputted tags as classes and attempt to classify the existence of each tag given the available information about the review. First, in Sections 4 and 5 we attempt to classify each tag independently of the others. Then in Section 6 we move to making a joint decision given the probabilities obtained from the independent classifications. We use the standard precision, recall, and balanced F-measure as performance metrics.

3.1 Datasets

Our datasets include a total of 3,245 reviews. We randomly split these reviews into training, validation, and test sets with an 80%, 10%, 10% split. For all experiments we performed training on the training set, used the validation set to find optimal parameters, and present results by applying the classifier with the optimal parameters from the validation set to the test set. We only include tags that occur at least 50 times, which amounts to 19 pros, 9 cons, and 8 affinities. Table 3.1 shows the tags and their frequencies for pros, cons, and affinities.

As one can see, the frequency of pros is much higher in comparison to cons. This causes many problems that we will discuss in later sections. Additionally, many of the tags have frequently occurring opposites, such as "Long Battery Life" and "Short Battery Life", and the affinities "Technically Challenged" vs. "Technically Savvy." We leverage these facts in Section 7 when optimizing the tag sets for each review. On the other hand, the distinction between several of the pros is quite vague. For example, "Reliable" and "Reliable Performance" can be interpreted to be the same attribute, as well as all of the pro tags that start with the word "easy." The existence of the con "None" is also a misnomer because the fact that there are no cons is not itself a con.

Table 3.1: Pro, Con, and Affinity Classes

Pros	
Frequency	**Class**
137	Accurate Maps
1,320	Acquires Satellites Quickly
2,051	Compact
223	Durable
143	Easy Controls
1,381	Easy Menus
141	Easy to Follow Directions
1,997	Easy to Read
1,967	Easy to Set Up
70	Easy to Use
242	Good Value
183	High Quality
396	Long Battery Life
144	Nice Visual Interface
167	Reliable
1,674	Reliable Performance
511	Secure Mounting
1,746	Simple Controls
433	Strong Construction
Cons	
Frequency	**Class**
278	Acquires Satellites Slowly
109	Bulky
131	Complicated Controls
175	Difficult Menus
86	Difficult to Read
224	Flimsy Mounting
344	Inaccurate
56	None
456	Short Battery Life

Table 3.1 Continued

Affinities	
Frequency	Class
600	Bad With Directions
1,096	Frequent Traveler
720	Gearhead
402	Outdoor Enthusiast
60	Practical
140	Professional
203	Technically Challenged
191	Technically Savvy

Each review contains a title, comment text, rating (from one to five), source (web or email), creation date (down to the second), author nickname, author location, length used, and bottom line. The source and creation date are not directly inputted by the consumer. The user inputs their author nickname and location by typing into two text boxes, so the information is not standardized and can include anything that the user decides to type. The bottom line is either "Yes, I would recommend this to a friend" or "No, I would not recommend this to a friend." The length used field indicates how long the consumer has used the product and has one of six possible values:

- Not specified
- One month or less
- One to three months
- Three to six months
- More than six months

3.2 Difficult Data

The majority of the reviews are very short, containing only a few sentences. Run on sentences with no punctuation are quite popular. In fact, the average number of sentences in the comment text is five while the average word count (counting punctuation as individual words) is 88.49.

This small amount of information makes the classification task rather difficult. For a moment, consider the review in Table 3.2a and attempt to assign the pros, cons, and affinities from Table 3.1. Even with the hint that the review contains one pro, one con, and one affinity, there is almost no chance that a human using any kind of reasoning skills would identify the correct tags. Perhaps you chose the pro of "Good Value" because the consumer said that the product was not expensive. You probably also thought that the con was "Short Battery Life" because of the mention of getting an extra battery. In fact, the pro that the consumer chose was "Easy To Set Up," the con was "Complicated Controls," and the affinity was "Outdoor Enthusiast." In our opinion, assigning these tags is nearly impossible given the limited information.

Table 3.2a: Review Example

Author	Harvey
Source	web
Creation Date	2007-04-14 16:38:27
Location	Chicago, IL
Length Used	Not specified
Rating	3
Bottom Line	Yes, I would recommend this…
Title	"I needed it"
Comment	"Nice to have to extra battery handy, never know when you will need it, and not expensive."

We found these kinds of issues to be frequent in our data. We posit that the fact that the user is asked to input tags tends to cause the user to mention other aspects of the product in their comment text. Indeed, many reviewers seem to not even bother to mention the pros and cons in their comment text when they have included them as tags.

Table 3.2b shows another review that is difficult to assign tags to. All remarks in the title and comments are negative and the rating is a one, yet the reviewer gave the product four pros and only one con, with the con not even specifically mentioned in the comment text. An open question not directly related to natural language processing that this brings attention to is whether the solicitation of the tags causes the reviewer to provide additional unique information or if it causes the reviewer to expend less effort constructing the comment text in favor of clicking the checkboxes next to the proposed tags.

Table 3.2b: Review Example	
Author	ht
Source	web
Creation Date	2007-08-01 16:47:25
Location	New Jersey
Length Used	Not specified
Rating	1
Bottom Line	No, I would not recommend this…
Title	"Don't buy....3 in a row no good"
Comment	"Went through 3 units in less than 1 week. Staples was very helpful but the problem is with the unit. Nextar customer service was of no help. Returned and got a Garmin Nuvi 200 and it is great. I should have went with a name brand to begin with. Be wary…"
Pros	Compact
	Easy To Read
	Easy To Set Up
	Simple Controls
Cons	Inaccurate
Affinities	Professional
	Outdoor Enthusiast
	Gearhead

4 Bag of Words

After an examination of the review data, we built a baseline bag of words Naïve Bayes classifier. We will not include a discussion of the technical details of the Naïve Bayes classifier, or any of the other machine learning algorithms we use, as they are not directly the point of this study.

Our bag of words features are standard unigrams. We experimented with both frequency unigram features and mere presence unigram features. We quickly found including frequency to be around 1.5 times better than using binary presence features, which is in contradiction to Pang and Lee's findings in sentiment classification but consistent with typical document classification results.

Due to the relatively fast training and testing time of the Naïve Bayes classifier (in comparison to SVMs, etc.), we were able to study the effect that various preprocessing methods have on performance. Our preprocessing methods include lowercasing all characters, removing all punctuation, and using various stemmers. Our stemmers include the Porter and Porter 2 Snowball stemmers [5], and the Lovins and Iterated Lovins stemmers [6]. We used the WEKA 3 library [7] for the classifiers in this section as well as the stemmers.

We also experimented with varying the vocabulary size in several ways, including dropping all words that have a frequency less than a given amount, removing the top X% of words ordered by frequency, removing the bottom X% of words ordered by frequency, and removing words based on the ratio of their occurrences in the positive class and negative class reviews given a particular tag. Our attempt at pruning the vocabulary based on ratio came as an attempt to remove words that may occur frequently, but that occur in both positive and negative reviews in equal proportion. (Here, when we say positive reviews we mean reviews that contain the particular tag that we are trying to reason about.)

In addition to the above preprocessing options, we also included normalizing the frequencies by the length of the sentence and scaling frequencies to [-1,1]. We introduced these additional preprocessing options primarily because we tried other classifiers such as SVMs which generally work better when the feature vectors are scaled to a particular range.

4.1 Naïve Bayes Results

We tried all unique combinations of lowercasing, removing punctuation, stemming, normalizing, and scaling to the range [-1,1], while also sweeping through the X% value in increments of five percent for removing the least frequent words in the vocabulary and for removing words with a ratio closest to one. Table 4.1 shows the results of these experiments when trying to classify the pro "Compact."

The first entry in the table shows the results without any preprocessing. The second includes, in order, lowercasing all words, removing punctuation, using the Iterated Lovins stemmer, and removing the least frequently occurring 30% of the vocabulary. Including these preprocessing steps improves the negative class F-measure by 8.6 points and the positive class F-measure by 5.3 points. Replacing the vocabulary reduction with a reduction that removes the 27% of the vocabulary that has a ratio between frequencies in positive and negative reviews closest to one improves scores further, edging out 0.7 more points in negative F-measure and 0.6 additional points in positive F-measure.

The bottom listing in Table 4.1 was the best performing Naïve Bayes bag of words classifier that we could muster. The Lovins stemmers almost always outperformed the Snowball stemmers, and stemming after lowercasing/removing punctuation consistently resulted in better performance than stemming first. In general, lowercasing improved scores more than removing punctuation. Normalizing and scaling improved scores in some situations, but did not appear in the best performing classifiers.

Table 4.1: Naïve Bayes Results on the Pro "Compact"

No Preprocessing			
Class	**Precision**	**Recall**	**F-measure**
Negative	36.7	33.3	34.9
Positive	76.1	78.8	77.3
Lowercase, Remove Punctuation, Iterated Lovins Stemmer, Remove Bottom 30% of Vocabulary			
Class	**Precision**	**Recall**	**F-measure**
Negative	51.0	37.9	43.5
Positive	79.0	86.5	82.6
Lowercase, Remove Punctuation, Iterated Lovins Stemmer, Remove 27% by Ratio			
Class	**Precision**	**Recall**	**F-measure**
Negative	53.2	37.9	44.2
Positive	79.2	87.6	83.2

4.2 Additional Baseline Classifiers

We also experimented with using other off-the-shelf classifiers such as various trees and nearest neighbor algorithms from WEKA, and SVMs with linear, polynomial, and radial basis function kernels using LibSVM [8]. All of the trees had vastly worse performance than the Naïve Bayes classifier. Despite following the LibSVM guide on parameter selection [9], we were unable to achieve performance with an SVM that was comparable to the best Naïve Bayes classifier.

5 Maximum Entropy Classifier

Our primary tool for our classification problem is a maximum entropy classifier. The maximum entropy classifier allows us to easily add many features to constrain the current data instance while leaving the rest of the probabilities pleasantly uniform (equally likely). We used the Stanford Classifier [10] as our out-of-the-box maximum entropy classifier.

In this section we will first describe our wide range of features. We will then discuss our hill climbing approach to finding the optimal feature set for a particular tag and present our optimal feature sets for several choice tags. We will end the section with the results of applying these optimal feature sets to the test set.

5.1 Features

Our features can be segregated into two groups: global features and textual features.

5.1.1 Global Features

Our global features are non-linguistic features that leverage the information that we have about the review/reviewer excluding the review comment and title. These features include most of the non-comment information described at the end of Section 3.1, such as the product's rating, the review's source, date/time of the review's creation, the author's username, the length used field, and the author's location. Most of these features are self-explanatory and only contain a few possible values.

By examining the training set we can easily see which features are most likely to be useful. For example, the product's rating is an obviously useful feature. Only 44% of reviews with a rating of one contain the pro "Compact," while 76% of reviews with a rating of five contain the tag. In the following subsections we will briefly elaborate on the date/time, username, and location features.

5.1.1.1 Review Creation Date/Time

The date and time that the reviewer submitted the review is stored in a standard format, so we can easily extract the month, day, year, hour, minute, and second of the submission. We include the month, day, year, and hour as separate features. We also include combinations, such as the year and month, year and hour, and month and hour. Additionally, we include the day of the week (Sunday through Saturday), the day of the month, the day of the year, the week of the month, the week of the year, and whether the review was submitted during A.M. or P.M.

We include many of these features because an analysis of the distributions obtained by conditioning on some of these features was significant. For example, a review submitted on Friday is far less likely to have a con in comparison with reviews submitted on Monday or Tuesday (with Monday being the most negative). Several of the features did not seem to be significant factors, but were simple to include and remove during our feature selection process if needed.

5.1.1.2 Username Features

We include the raw username as a feature out of the off-chance that we come across multiple reviews by the same reviewer. We also include the amount of characters in the username as a feature, and we make a crude attempt to break usernames into separate words. To break usernames into separate words, we first tokenize the username by spaces (some users are kind enough to separate their username into words for us). We also separate the username into words by splitting whenever we see a change in capitalization or when we see a change from alphabetical characters to numbers or symbols. We use each of the words derived from the username as separate features.

Our reasoning behind these username features is that many usernames are indicative of the personality or type of person that wrote the review, which is most helpful for classifying affinities. For example, our data contains the follow helpful usernames: The Traveler, SoccerMom, Old timer, UCBearCatMike, and The Tech Guy. All of these usernames provide hints that are sometimes even more useful for learning about the reviewer than the comment text itself since many reviews do not contain any personal information in the comment text.

5.1.1.3 Location Features

We include location features for the author's street, city, state, region, and country, as well as features indicating whether any of the previously listed location values are present in the review information. The intuition behind including these features is that reviewers from the same state may on average have similar affinities and may be more likely to complain about certain cons or praise certain pros that are important in their area. As a reminder, the user inputs their location into a textbox, so they are allowed to type any free-flowing text that they desire. Thus to extract information such as the city and state of the author, we must identify which segments of location field denote the city, state, etc.

We identify the various location attributes using modified gazetteers from GATE [11]. These gazetteers contain popular cities, street names, and various spellings of the U.S. states. We do not make any attempt to merge multiple spellings of states or cities. Thus while "CA" and "California" are both recognized as states, they are not grouped into the same feature.

The gazetteers work surprisingly well, but many location strings remain with unknown location types for many words. We employ a simple procedure to fill in the missing values.

We start at the last token in the string and work towards the front. If we come upon a token with a missing location type, we look at the location type of the token one to the right and assign a location type according to Table 5.1.1.3.

After assigning a location type to all tokens we chunk the tokens together based on their type. This process, which allows us to treat multi-word states and cities as single entities, involves joining neighboring tokens that have the same location type.

Table 5.1.1.3: Assigning Location Types

Previous Location Type	Assigned Current Type
State	City
City	City
Region	Region
Country	State
Street	Street

5.1.2 Textual Features

Our textual features include many basic features such as the amount of words in the title, the number of sentences in the comment, the amount of words in each sentence, and whether the title contains a question mark. We also include choice n-grams. At first, we experimented with including all unigrams, and then added bigrams and trigrams. We found that including all bigrams and trigrams hurt performance. We also found that selecting only specific unigrams (as well as larger n-grams) increased performance.

Thus we include as features the unigram, bigram, trigram, and 4-gram that begin and end each sentence (including the title). We also include all adjectives, with the comparative, superlative, and normal adjective tags distinguished separately. Additionally, we include each word with the previous word's part of speech tag as a feature and if the previous word was an adjective, then we include as a feature the bigram containing the previous word and the current word.

We also include features pertaining to tag tokens. We split into tokens the particular tag that we are currently making a classification decision on. We then look for n-grams of the tokens in the title and comments, including these n-grams as features. Upon finding an n-gram of tag tokens, we also include the previous word joined with its part of speech tag as a feature. We continue adding these word + part of speech features for up to three words in both directions. We also include the closest verb and closest adverb to the left of the n-gram of tag tokens as features, and include features indicating which tag tokens were found and the frequency of the tokens. We did not construct any lists of synonyms or similar phrases for each tag due to time constraints.

5.2 Feature Selection

We select the optimal feature set by performing a random-walk hill climbing search through the feature space. We select optimal feature sets individually for each tag, and also include using the top ten sets of preprocessing options from our study with the baseline system. In this section we present the results for select tags, such as our running example of the pro "Compact." As usual, we performed our feature selection based on the scores from the validation set, and in the results section (Section 5.3), we present the scores for applying the feature sets to the test set.

5.2.1 "Compact" Feature Set

The highest performing preprocessing options set was lowercasing all words and then using the Porter Snowball stemmer to stem the words. Of the global features, the rating, source, length used, and author's street, city, region, and country were useful. The following date/time features were found to be important: year, month, year + month, year + hour, month + hour, day of week, day in month, and whether the review was submitted during AM or PM hours. The author's username split into tokens as well as a special tag indicating the last token in the username were helpful.

Among the textual features, the tag token features were most prominent. The count of each tag token, the part of speech of the word immediately to the right of the tag token n-gram, and the previous adverb were useful. Out of the three word + part of speech features to either side of the tag token n-gram, only the feature two to the left, and the features two and three to the right were used.

The only title-specific feature that our feature selection found useful was the number of words in the title while the only comment-specific feature was the total word count. The only other textual features were the unigram ending each sentence, the trigram starting each sentence, and the previous word + current word when the previous word's part of speech tag was an adjective.

An interesting result is that the majority of the features are not textual, although we posited that the textual features were playing an important role in the classification. To determine which groups of features mattered most, we selectively removed groups of features and applied the resulting classifier to our data. Table 5.2.1 details the results, listing the group of features removed and the amount of decrease in sum of negative class and positive class F-measure.

To our surprise, while the textual features definitely play a role, the length used and date/time features are far more important. Removing all date/time features caused an 8.62 drop in F-Measure while removing length used caused an 8.65 drop and removing all textual features only caused a 6.57 drop! While it is good to see that the global information significantly contributes the classification task, we have exhausted most of the features that can be derived from this information. Thus we see any and all future improvements coming from more sophisticated textual features.

5.2.2 "Bulky" Feature Set

To contrast the feature set selected for the "Compact" tag, we examined the feature set selected for its opposite – the con "Bulky." The "Bulky" feature set exhibits many differences from "Compact." The rating, length used, title length, and comment sentence word count features are still used, but the source and author name features are not. Of the author location features, only the country is used. Among the date/time features, only the day of the week and day of the month are used.

Both the previous verb and previous adverb for tag tokens are used whereas "Compact" only used the previous adverb. Additionally, the search found all three word + part of speech tags in both directions from a tag token n-gram to be helpful. None of the n-grams beginning sentences were found to help, but the unigram and bigram ending the sentence improved performance.

Table 5.2.1: Importance of Features

Removed Feature Group	Change in F-Measure
All tag token features	-1.24
Previous word + current word given the previous word was an adjective	-1.22
Trigram beginning sentence and unigram ending sentence	-2.29
Rating and source features	-4.75
Author name features	-0.30
Date/time features	-8.62
Length used	-8.65
Location features	-0.78
All textual features	-6.57

5.3 Results

In this section we present the results of using our classifier to classify for each tag independently of the other tags. Due to time constraints, we were unable to examine many different tags while developing our features. We developed all of the previously listed features through an in-depth analysis of the "Compact" tag. As a result, our performance on this tag is quite good. Most of our features tend to carry over well to other pro tags, but the performance on cons and affinities is dismal in comparison.

Table 5.3a presents our best, second best, worst, and cumulative performance on pro tags. The worst performing pros tend to be tags that are least frequent, occurring around

15% of the time. The best performing pros have a much more balanced frequency, occurring in about 60% of the reviews.

Table 5.3a: Best, Second Best, Worst, and Cumulative Performance on Pros

Best: "Compact"			
Class	**Precision**	**Recall**	**F-measure**
Negative	82.2	69.3	75.2
Positive	82.7	90.8	86.6
Second Best: "Easy To Read"			
Class	**Precision**	**Recall**	**F-measure**
Negative	74.4	73.1	73.7
Positive	85.2	86.0	85.6
Worst: "Long Battery Life"			
Class	**Precision**	**Recall**	**F-measure**
Negative	90.7	95.3	92.9
Positive	36.4	21.6	27.1
Cumulative			
Class	**Precision**	**Recall**	**F-measure**
Negative	91.5	92.0	91.8
Positive	74.4	73.1	73.7

Table 5.3b displays the cumulative results for cons and affinities, as well as overall results that sum all tag decisions together. The cons and affinities seem to suffer most from their lower frequencies. For many of these tags the classifier decides to classify all reviews as not containing the tag.

Table 5.3b: Cumulative Performance on Cons, Affinities, and All Tags

Cumulative on Cons			
Class	**Precision**	**Recall**	**F-measure**
Negative	94.2	99.0	96.6
Positive	55.6	17.2	26.2
Cumulative on Affinities			
Class	**Precision**	**Recall**	**F-measure**
Negative	89.9	95.5	92.7
Positive	50.5	29.8	37.5
Cumulative on All Tags			
Class	**Precision**	**Recall**	**F-measure**
Negative	91.9	94.8	93.3
Positive	70.9	60.3	65.2

6 Multi-Tag Optimization

To increase performance, we developed two methods that attempt to perform multi-tag optimization to find the best tag set for a given review. Our first technique is a classifier built on top of the maximum entropy classifier's output and our second is a "probabilistic" combinatorial search algorithm. Both methods use the probabilities that our maximum entropy classifier outputs for each tag on each review.

6.1 Second-Layer Classifier

Our second-layer classifier uses as features the probability of the positive class for each tag as reported by our maximum entropy classifier. Thus, there are 36 features for each review. Due to the small number of features and the ease at which one can "plug-in" classifiers using WEKA, we were able to try a multitude of classifiers for the task, including Naïve Bayes, logistic regression, nearest-neighbor variants, SVMs, and various trees. We found the standard nearest-neighbor classifier using normalized Euclidean distance to perform best.

Our results are somewhat lackluster. When using all 36 probabilities as features, we were unable to obtain higher scores on cons or affinities, and we were only barely able to edge out a better score on the pros. Table 6.1 presents these results.

Table 6.1: Cumulative Performance on Pros, Cons, and Affinities using all 36 Features

Cumulative on Pros			
Class	**Precision**	**Recall**	**F-measure**
Negative	91.8	91.7	91.8
Positive	74.0	74.1	74.1
Cumulative on Cons			
Class	**Precision**	**Recall**	**F-measure**
Negative	94.2	98.9	96.5
Positive	52.2	17.2	25.8
Cumulative on Affinities			
Class	**Precision**	**Recall**	**F-measure**
Negative	89.1	94.1	91.5
Positive	47.8	31.8	38.1

We believe that the method would have more success if our classifier's performance on cons and affinities was better. We found only marginal improvement by excluding various sets of tags from the features. In particular, the positive F-measure for affinities increases to 42.2 when only using the affinity probabilities as features, but its negative F-measure was only 91.6, which is still below the classifier's score. The pro and con scores were highest when using all 36 features.

6.2 Combinatorial Search

After a dissatisfying first try with our second-layer classifier we changed routes and developed an algorithm that attempts to find the optimal tag set given various probability distributions derived from the training data. In this section we will describe the probabilities that we include, our scoring metric, our two search methods, and our method for finding the best weights to use in the scoring metric.

6.2.1 Probability Distributions

Our first group of probabilities that we include are the probability of a tag set size conditioning on various elements. These probabilities include conditioning on nothing as well as conditioning on each tag in the set, and the review's rating, author's state/province, length used field, source, comment word count, sentence count, title word count, bottom line, day of week, hour of day, and month.

Our second group of probabilities pertain to the frequency of occurrence of each tag set. Again, these probabilities include conditioning on nothing as well as conditioning on each of the items listed for the tag set size probabilities. We efficiently compute these probabilities through the use of bitsets. Before beginning the search, we compute two bitsets for each individual tag, where the number of bits is the amount of reviews in the training set. For the first bitset, each true bit indicates that the review referenced by the bit's index contains the particular tag. The second bitset captures the opposite, containing a true bit for each review that does not contain the tag.

To compute the probability of a particular tag set, we simply apply logical AND to each tag's appropriate bitset (depending on whether the tag exists in the tag set or not), and divide by the total number of reviews in the training set. We apply a small amount of smoothing so that no probabilities are zero. To compute the probability conditioning on some element such as the review's rating, we compute bitsets for each possible value of the conditioning element and include the appropriate bitset in the logical AND. Instead of dividing by the number of training reviews, we divide by the number of reviews that satisfy the condition. We apply bucketing to deal with conditioning on information such as comment word count.

6.2.2 Scoring Metric

Our scoring metric that we are trying to maximize is a weighted sum of the classifier's probability and the previously described probabilities from the training data. We call our algorithm a "probabilistic" combinatorial search algorithm because we make no attempt to ensure that the score is a true probability – and indeed it is not in our formulation.

6.2.3 Search Methods

We employ two search methods: an almost-complete search and an even less complete probing method. In the following descriptions, let N be the number of tags that we are considering. When attempting to optimize for all tags, N is 36.

6.2.3.1 Almost-Complete Search

Our almost-complete search uses a priority queue to manage tag sets. We begin by adding 2N incomplete tag sets to the queue. The first N of these tag sets have one of the tags set to true and all other tags set to unspecified (we have not made a decision on them yet). The second N are the opposite, having one of the tags set to false and all others unspecified.

We then repeatedly remove the highest scoring item from the queue. If the highest scoring item contains no unspecified tags, then we return the tag set as the optimal tag. This tag set is the optimal tag set because specifying additional tags only adds more restrictions, and thus always lowers the score.

If the tag set contains unspecified tags then we loop over these unspecified tags and add two new tag sets to the queue for each unspecified tag. The first of these tag sets includes the particular unspecified tag assigned to true while the second has the tag assigned to false. The algorithm is "almost-complete" because we use a bounded priority queue of 1,000,000 tag sets.

6.2.3.2 Probing

The almost-complete search method can handle tag sets of sizes up to 10 or 12. Unfortunately, this is a far cry from our goal of optimizing all 36 tags simultaneously. In our probing method we collect anywhere from 1,000 to 18,000 unique probes and return a k-best list of tag sets gathered during our probing.

For each probe, we begin with all tags unspecified. We generate a random permutation of the tags, which we use as the ordering that we will assign the tags. When making each tag decision, we have the option of assigning the tag as true or false. We determine which value to assign by calculating the score of both resulting incomplete tag sets and using the score to calculate a probability of choosing true or false. In particular, if the score of choosing true is X and the score of choosing false is Y, then we choose true with a probability of X / (X + Y). Our algorithm quits and returns the current k-best list if it fails to find a new unique probe after 50,000 attempts.

6.2.4 Weights Optimization

The probability distributions that we use in our scoring metric results in needing to choose 25 weights. After trying various methods for assigning weights, we found the best approach was a stepwise search that gradually adds more nonzero weights into the scoring metric. We limit the weights that each distribution can take on by only allowing each distribution to hold equal weight or less weight in comparison to the classifier's probabilities.

We begin with all weights set to zero except the classifier's probabilities. We then attempt to assign weights to each of the distributions one at a time, keeping track of the assignment that results in the highest score. After iterating over all distributions, we have the best scoring weight set which contains nonzero weights for the classifier's probabilities and only one other distribution. We then continue recursively, attempting to assign weights to each zero weight and keeping the nonzero weights from the previous round fixed. After the second iteration, we have the best scoring weight set that contains nonzero weights for the classifier's probabilities and two other distributions. If completing an iteration does not increase the score over the previous iteration then we quit.

We try weights for a distribution using a binary-style search. We first set the end points for the possible weight range to zero and the value of the classifier's weight. We then run the combinatorial search algorithm using the weight that is half-way in-between the two end points. If the resulting score is higher than our previous best score then we move upwards into searching in the top half of the range, and down to the bottom half otherwise. We add one additional trick in that if moving upwards at any point does not produce a better score then we return and attempt to search the bottom half of the particular range at that split point. We do not include a similar procedure when searching the bottom half of a range.

6.2.5 Weights Optimization Results

As the next section will show, we found our best performance gains for affinities by jointly optimizing only the affinity tags (not including pros or cons in the optimization process). In this section we present the weights that we found to be optimal for the affinity tags. Table 6.2.5 shows the nonzero weights that improved the score over solely using the classifier's probabilities.

Aside from the classifier itself, the strongest weights were given to the set size probabilities conditioned on rating, source, and author location. One can easily find a reason why all of the probability distributions with nonzero weight might help improve the classification, especially since we are considering affinities. An interesting outcome is that different types of distributions received nonzero weights depending on the type of tags that we tried to jointly optimize. In the case of optimizing all pros together, nonzero weights were given to distributions pertaining to the month, day of week, and bottom line, whereas here with affinities information such as the source and author location is important.

Table 6.2.5: Optimal Weights for Affinities

Probability Distribution	Weight
Maximum Entropy Classifier	100
Occurrence Given Rating	1.6
Occurrence Given Length Used	31.3
Occurrence Given Word Count	50
Set Size Given Rating	82.8
Set Size Given Length Used	25
Set Size Given Source	75
Set Size Given Author Location	75
Set Size Given Sentence Count	26.6

6.2.6 Results

Our combinatorial approach offers great improvements in comparison to our underperforming second-layer classifier, but the process is also vastly more computationally expensive. As a result, we were unable to attempt optimizing all tags simultaneously, and the optimization for larger tag sets is less accurate since we had to cut the optimization short due to time constraints.

Table 6.2.6a shows the results of optimizing all 8 affinities simultaneously. For ease of comparison, we include the previous results for the maximum entropy classifier and the best affinity result for the second-layer classifier. While our performance on the negative class decreases, the performance on positive class more than makes up the difference.

Table 6.2.6a: Jointly Optimizing Affinities

Combinatorial Optimization Result			
Class	Precision	Recall	F-measure
Negative	91.3	95.0	93.1
Positive	55.0	40.3	46.6
Maximum Entropy Classifier Result			
Class	Precision	Recall	F-measure
Negative	94.2	99.0	96.6
Positive	55.6	17.2	26.2
Second-Layer Classifier Result			
Class	Precision	Recall	F-measure
Negative	89.8	93.4	91.6
Positive	48.8	37.1	42.2

For all other results in this section we limited our tag sets to consider only the 3 most frequent affinities, 4 most frequent cons, and 7 most frequent pros. Most likely due to our classifier's poor performance on cons and affinities in general, our combinatorial optimization improved performance on pros the most when only considering the pros by themselves. Table 6.2.6b shows these results for optimizing the 7 most frequent pros as well as the maximum entropy classifier's cumulative performance on the 7 pros.

Table 6.2.6b: Jointly Optimizing Pros

Combinatorial Optimization Result			
Class	Precision	Recall	F-measure
Negative	93.2	89.8	91.5
Positive	71.3	79.5	75.2
Maximum Entropy Classifier Result			
Class	Precision	Recall	F-measure
Negative	91.5	92.0	91.8
Positive	74.4	73.1	73.7

Our combinatorial optimization performed best on cons when considering the pros, cons, and affinities together. Table 6.2.6c shows these results. While the negative F-measure decreases slightly, the positive F-measure increases by 3 points.

Table 6.2.6c: Jointly Optimizing Pros, Cons, and Affinities

Combinatorial Optimization Cons Result			
Class	Precision	Recall	F-measure
Negative	91.3	97.6	94.4
Positive	56.2	24.7	34.3
Maximum Entropy Classifier Cons Result			
Class	Precision	Recall	F-measure
Negative	91.0	98.2	94.5
Positive	59.6	21.2	31.3

7 Future Work

While we tried to cover as much ground as possible, many avenues for improvement remain. One helpful study needed for the evaluation process is a human study to see how humans perform at making these tagging decisions, especially in light of the difficult data issues discussed in Section 3.2. We also leveraged only rough textual features. More improvements could be realized by using phrase structure trees or dependency trees to indentify which words modify other words, heads of phrases, etc. We also made no explicit attempt to model negations, which are important.

Additionally, we used Balie [12] for tokenization and sentence boundary detection. Unfortunately, we found its performance to be less than desirable as it always split words with hyphens into separate words and made odd decisions for many sentence boundaries. Including better tokenization and sentence segmentation algorithms would likely improve overall performance.

Lastly, we spent all of our maximum entropy classifier feature development time studying only a few pro tags. This clearly resulted in poor performance on cons and affinities in comparison to pros. By dedicating additional time towards examining the specific difficulties in con and affinity tagging we believe we can achieve much better performance, which would also carry over to more substantial improvements in our combinatorial optimization algorithm because the con and affinity information would be of higher quality.

8 Conclusions

In this paper we presented various methods for assigning pro, con, and affinity tags to reviews. Our approach is different from previous approaches in that we consider a fixed tag lexicon for each tag type, which we believe is more suitable to real-word applications as aggregate data can more easily be presented to users in comparison to extracting snippets of text that contain a pro/con from each review. Our maximum entropy classifier presents strong performance over our baseline bag of words Naïve Bayes classifier, with 75.2 negative class F-measure and 86.6 positive class F-measure vs. the 44.2 negative class F-measure and 83.2 positive class F-measure achieved by the best performing preprocessing methods for Naïve Bayes on the "Compact" tag. Our multi-tag optimization methods show improvements over the maximum entropy classifier, but are limited by the relatively poor quality of the maximum entropy classifier's con and affinity probabilities. While we covered many techniques and included many features in our study, a vast amount of opportunities remain.

9 Acknowledgements

We thank Robert Chea and Joshua Greenough of PowerReviews for the product review data, Professor Manning for the insightful initial discussions about our project, and Pichuan Chang and Paul Baumstarck for constructing the list of resources about various parsers, taggers, and other NLP-related libraries. Todd Sullivan is supported by an NDSEG Fellowship sponsored by the DOD and AFOSR.

10 References

[1] Bo Pang , Lillian Lee and Shivakumar Vaithyanathan, "Thumbs up? Sentiment Classification using Machine Learning Techniques," Proceedings of EMNLP 2002.

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

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

[4] Kim, S-M. and E.H. Hovy, "Automatic Identification of Pro and Con Reasons in Online Reviews," Poster. Companion Proceedings of the conference of the ACL, Sydney, Australia, 2006

[5] M.F. Porter, Snowball. http://snowball.tartarus.org

[6] Julie Beth Lovins, "Development of a stemming algorithm," Mechanical Translation and Computational Linguistics, 1968 11:22-31.

[7] Ian H. Witten and Eibe Frank, "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
[8] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM : a library for support vector machines," 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
[9] C.-W. Hsu, C.-C. Chang, C.-J. Lin., "A practical guide to support vector classification" (http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf)
[10] Christopher Manning and Dan Klein, "Optimization, Maxent Models, and Conditional Estimation without Magic," Tutorial at HLT-NAACL 2003 and ACL 2003.
[11] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications," Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). Philadelphia, July 2002.
[12] David Nadeau, "Bailie – Baseline Information Extraction: Multilingual Information Extraction from Text with Machine Learning and Natural Language Techniques," 2005. (http://balie.sourceforge.net)