CS 224N Final Project:
Automated extraction of product attributes from reviews
Nikhil Gupta, Praveen Kumar, Rahul Gupta
June 5, 2009

Abstract - Over the past few years, there has been huge growth on the internet in the field of online reviews of products. These reviews serve as means of assessing the general outlook of a product to the potential buyers. Due to the very “informal” and unstructured setting in which these reviews are obtained, each review might not consider all the features of the product. Also, with increasing number of reviews for each product, it becomes difficult for the user to read through and identify all the key features to consider while buying a product. This paper aims at providing a method for finding the key features of products by looking at a number of reviews of the same product.

1. Introduction

Information extraction from unstructured text is one of the challenges being tackled by current research in the field of Natural Language Processing and Data Mining. In the field of information extraction from product reviews, most of the work has focused on finding the values for a set of pre-defined attributes. Recently, there has been growing interest in the automated learning of the attributes themselves, and then finding the associated values. Our goal is to use the language structure of a sentence to determine if a word is a feature in the sentence. For this we describe our approach in which we first run a POS tagger on the reviews data, and then generate input vectors using these POS tagged reviews. We formulate the problem of extracting features as a classification problem, where given a word, the goal is to classify it as a feature or not-feature. We tried a number
of classifiers, and report our results and error analysis on them. The metrics we use to access the performance of our system are the standard precision, recall, and balanced F measure.

2. Previous Work

One of the initial papers in the field of “attribute” extraction by Hu et al [1] talks about using a frequency based approach to identifying the features in product reviews. They order the noun-phrases by frequency and then have different manually defined settings to find the features (like lower cutoff, upper cutoff etc). Though they are able to achieve a good workable system with these methods, their assumption that a feature would always be a noun is not always true. There can be multi word features like “optical zoom”, “hot shoe flash” where one of the words is an adjective. They take a more holistic approach to the problem and use the opinion (sentiment) words to find infrequent features.

In [2], Ghani et al have shown success in attribute-value pair extraction using co-EM and Naïve Bayes classifiers. However, their work focused on official product description from merchant sites, rather than on reviews. As these descriptions generally have a very particular way of listing the details (like “Operating system: Windows Vista Home”), performing this task is different from working with unstructured user reviews.

Popescu et al’s OPINE system [3] also uses the dataset provided in [1]. They explicitly extract noun-phrases from the reviews (with a frequency based cutoff) after POS tagging and then compute Pointwise Mutual Information scores between the phrase and meronymy discriminators associated with the product class. This again assumes that features are always nouns and misses out on features which are not nouns or are combination of different POS tags.

3. Dataset

```
1 [c] I glad to own .
2 # I have had this phone for about 5 months .
3 battery+[+]## I treat the battery well and it has lasted .
4 ## I at my heaviest usage , I must recharge after 3 days .
5 ## It lasts about 8 days otherwise and has lasted up to 10 when I was making very few calls .
6 ## Signal strength will affect the battery life .
7 ## frequent signal searches eat up battery power .
```

Figure 1. A sample review from the data set used. This is for the “Nokia 6610” product.

We used the annotated product review data provided by Hu et al [1] to build and test our system on. This dataset contains a total 314 reviews for 5 different electronic products crawled from Amazon.com. Each product’s reviews are provided in a flat text file, and each review consists of a title and associated review text. Some basic preprocessing has been done, and each line of the review is on a separate line in the text file. The feature being talked about is indicated on a per sentence
basis along with a numeric positive/negative sentiment value about the feature. The features could be longer than a single word, for example “battery life”.

We had to do some clean up of the data before we could run the POS tagger on it. This included work like fixing lines not ending with a full stop, removing ellipsis (…) etc. Most of this work was done to ensure that we could maintain a one-to-one correspondence between the raw data provided and the data we generated after POS tagging using the Stanford Tagger.

To mark a word in the sentence as a feature, we compared it with the set of annotated features for that line. Some features were not annotated in their exact form. For example “megapixels” was converted to “megapixel” in the annotation. This led to this word not being marked as a feature in the sentence. To overcome this problem, we did stemming of both the sentence words and the features annotated in that sentence, before marking feature/not feature.

After running the Stanford POS tagger on the data, we generated three sets of data files. First, the initial feature set for each word consisted of the “relative” frequency of that particular word in the reviews for that product, its stemmed form, POS, and if that word was marked as feature or not in the original data. We also determined if the word had associated “attributes” in WordNet when the word was used as a noun, and appended that to the data.

<table>
<thead>
<tr>
<th>Product</th>
<th>Product category</th>
<th>Number of reviews</th>
<th>Average lines/review</th>
<th>Unique Features annotated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon G3</td>
<td>Electronics - Digital Camera</td>
<td>45</td>
<td>14</td>
<td>99</td>
</tr>
<tr>
<td>Apex AD 2600 Progressive-scan DVD player</td>
<td>Electronics - Multimedia Device</td>
<td>99</td>
<td>8</td>
<td>109</td>
</tr>
<tr>
<td>Creative Labs Nomad Jukebox Zen Xtra 40GB</td>
<td>Electronics - Multimedia Device</td>
<td>95</td>
<td>6</td>
<td>170</td>
</tr>
<tr>
<td>Nikon Coolpix 4300</td>
<td>Electronics - Digital Camera</td>
<td>34</td>
<td>11</td>
<td>71</td>
</tr>
<tr>
<td>Nokia 6610</td>
<td>Electronics - Mobile Phone</td>
<td>41</td>
<td>44</td>
<td>106</td>
</tr>
</tbody>
</table>

Table 1. Overview of the the dataset.

The second dataset we created marked a word as feature if it was marked as feature anywhere in the reviews for that product. This was both good and bad. The benefit we got from this approach was that we marked feature like “quality” in places where the annotator had missed them. On the flip side, we ended up marking words like “picture” as feature where it was not used in this sense.

The third dataset was an input file for the WEKA-3 classifier [5] that we used to rapidly test classifiers on the dataset. We generated a feature set of POS of window size 5 around each word, its relative frequency, distance of the nearest adjective from the word.
3.1 Problems with the dataset

Since we are using a hand annotated dataset for our training and testing purposes, the results of our system depend highly on the quality of the hand annotation done. After some analysis of the dataset we observed three kinds of errors in the hand annotation.

- Inconsistency – Some words are marked as feature in some of the sentences, but in others they are not. For example “photo quality” is marked as a feature in some lines in Canon G3 reviews set, but in some others it is not.
- Incomplete annotation – Some features are completely missed in the annotation, for example “shutter speed” is a feature that is talked about in multiple reviews in Canon G3 reviews set, but it is not marked as a feature anywhere.
- Wrong annotation – Some words that should not actually be marked as features are marked as features. This mostly occurs when the value of a feature is marked. For example in case of Canon G3 reviews dataset, “8mb card” is marked as a feature, which we consider as wrong annotation. Here, “8mb” is a value for the “memory card” feature, and it should not have been marked.

4. System Design

Our first attempt at solving this problem was based on the intuition that a product attribute is usually a noun phrase, and that people generally talked about a small set of common attributes of the product. This approach did not perform well. We then dropped the notion that a feature is always a noun, and tried using a generic Naïve Bayes classifier. Finally, we tried the EM and MaxEnt classifiers to build a probability model from the data.

4.1 Basic Implementation

Our first approach for extracting features was based on the intuition that product features are usually nouns or noun phrases [1]. Using the POS associated with words during the data generation phase, we created a new view of data where each sentence in the review was considered as a bag of words. The words chosen to represent a sentence were those that were marked as nouns (NN/NNS). We chose to ignore proper nouns, which we believe cannot be features associated with a product. Furthermore, as we observed in the data, some phrases that represent features (such as optical zoom) were made of two classes of words, nouns and adjectives. So as to be able to detect such features, we also included words marked as adjectives (JJ/JJR/JJS) in our bag of words model.

The next step was to extract frequent features from the candidate feature words. We used an implementation based on the APRIORI algorithm [10] for identifying frequently occurring word/word pairs from the bag of words data model. Using a support threshold of 0.5%, we got a good set of candidate features. However, we observed that because of inclusion of adjectives when finding the frequent item sets, we got many candidate features that actually were opinion words (like
good, best, bad). In order to filter out such frequent items, we considered only those single items that occurred as a noun somewhere in the corpus, or in case of word phrases, if at least one of the word in the phrase occurred as a noun somewhere in the corpus. The resulting set of words was our feature set for the product trained on.

4.1.1. Results
Our basic system performed rather poorly as seen from the precision and recall values we got (Table 2). Analyzing the set of features extracted by our base system, we found many proper nouns (such as Olympus, Nikon for the Canon data set) and opinion words (such as best, good, great) being marked as features. We manually removed these types of features from the features extracted by the basic system, to see if incorporating some domain knowledge into our algorithm had any impact on the efficiency. Table 3 shows the feature words that we removed for the Canon data set, and the results we got with pruned set of features are shown alongside the original results in Table 2. Though the results improved slightly, still we did not get results good enough to be useful as inputs for an Information Extraction system.

Analyzing the reason for such poor performance, we found that our system was marking all occurrences of a feature as feature, while in the annotated dataset we are using, this was not the case (as mentioned in section 3.1). Also, the annotation in the dataset had missed certain features (like flash card in Canon reviews), but which our system was correctly classifying. Even though this means that our system is good in certain aspects, the measures of performance we were using was not indicative of the system goodness. Therefore, we found out the precision and recall of our system only for the “feature” class, and these figures were much better (Table 2) than what we had obtained using the standard precision/recall metrics.

<table>
<thead>
<tr>
<th></th>
<th>Original Extracted Features</th>
<th>Manually Pruned Features</th>
<th>Feature Precision</th>
<th>Feature Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>Apex</td>
<td>13.688</td>
<td>79.803</td>
<td>14.74</td>
<td>38.47</td>
</tr>
<tr>
<td>Canon</td>
<td>13.497</td>
<td>81.686</td>
<td>13.44</td>
<td>35.67</td>
</tr>
<tr>
<td>Creative</td>
<td>12.307</td>
<td>78.448</td>
<td>11.44</td>
<td>35.33</td>
</tr>
<tr>
<td>Nikon</td>
<td>16.028</td>
<td>89.328</td>
<td>12.74</td>
<td>34.43</td>
</tr>
<tr>
<td>Nokia</td>
<td>16.146</td>
<td>86.402</td>
<td>16.1</td>
<td>43.16</td>
</tr>
</tbody>
</table>

Table 2. Results of the basic approach.

<table>
<thead>
<tr>
<th></th>
<th>Amazon, Best, Camera, Canon, Olympus, Powershot, S330, More, Most, Nikon, G3, Good, Great, G2</th>
</tr>
</thead>
</table>

Table 3. Filtered words for Canon dataset.
For the Canon dataset, this system was able to extract features like "flash card", "megapixel", "optic viewfinder", "shutter speed", "exposure control", "auto exposure setting", "Lcd screen" and "focus mode". However, features that are discussed rarely, like "dial", "diopter adjustment dial", "spot meter", "lens cap" and "prints" were misclassified by our system. This is because we consider only words that are frequent as potential features, and hence miss infrequently occurring features. Another shortcoming of this basic system was that for phrase features like "picture quality", it was classifying "picture" as a feature in phrases like "took their picture with their camera". Clearly, identifying whether a particular feature is a phrase feature as opposed to a single word feature would help improve the performance of this baseline system.

4.2 Naïve Bayes

After trying the initial approach, we used the Naïve Bayesian classifier (in WEKA) for classification. The features generated for this were the POS tags of a window of 5 around the word, relative frequency for each word in the document and distance of the nearest adjective. This gave better results than the initial approach we had taken, as we are now building a probabilistic model of the data. However, we found the results to be not so encouraging. Consider the fact that a feature mostly occurs in the form JJ NN WRB ("..good flash which.."), since Naïve Bayesian classifier considers the probability of the POS of the previous word independently of the POS tag of the next word (independence assumption), it misses out on picking that this particular grammatical form has a very high occurrence probability for the FEATURE class. Also, due to the same reasons, it gives an unacceptably large number of false positives. Thus, we get a high recall, but a very low precision of around 10-15%, as indicated in Table 4.

<table>
<thead>
<tr>
<th>Training set</th>
<th>Test set</th>
<th>Feature Precision</th>
<th>Feature Recall</th>
<th>Feature FB1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>Canon</td>
<td>0.172</td>
<td>0.677</td>
<td>0.274</td>
</tr>
<tr>
<td>Canon</td>
<td>Nikon</td>
<td>0.174</td>
<td>0.794</td>
<td>0.286</td>
</tr>
<tr>
<td>Canon</td>
<td>Apex</td>
<td>0.114</td>
<td>0.478</td>
<td>0.184</td>
</tr>
<tr>
<td>Canon</td>
<td>Canon</td>
<td>0.51</td>
<td>0.831</td>
<td>0.632</td>
</tr>
<tr>
<td>Canon</td>
<td>Nikon</td>
<td>0.386</td>
<td>0.864</td>
<td>0.534</td>
</tr>
<tr>
<td>Canon</td>
<td>Apex</td>
<td>0.562</td>
<td>0.412</td>
<td>0.806</td>
</tr>
</tbody>
</table>

Table 4. Results of Naïve Bayes classification.

We further analyzed the data and found that for words classified as “features”, the POS tags of the window of 5 words were the best indicators. Some analysis results of these POS tags are given in Table 5. We observed that many sentences that discussed features clearly fit this frequent pattern. For example consider the sentence describing the feature “flash”: “This camera has a very good flash which works during bright day light as a fill in”. Here very is tagged as superlative adjective, good is an adjective, flash is noun and which is tagged as a wh-adverb.

<table>
<thead>
<tr>
<th>Previous-Previous word</th>
<th>Previous word</th>
<th>Current word</th>
<th>Next word</th>
<th>Next-Next word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adverb, comparative</td>
<td>Adjective</td>
<td>Noun</td>
<td>Verb, present tense</td>
<td>no patterns</td>
</tr>
</tbody>
</table>
4.3 Bayes Formulation and EM

The idea here is to find a probabilistic estimation of a word being classified as a product feature based on its part of speech, word frequency, the word itself. As from the data we can see that words with a certain frequency (<50) and words with certain Part of speech (like NN) and finally words themselves play a very key role in identifying a feature. We don’t add any context information here, which we’ll do in Maxent (next section). We make the assumption the probability that a word gets classified as a feature based on its POS is independent of it being classified as feature based on other attributes like frequency and stemmed word. The Bayes net formulation is as shown in the Figure 2.

Figure 2. Bayes-EM model.
\[ P(Y, P, W, F) = P(Y) \cdot P(Y|W) \cdot P(Y|P) \cdot P(Y|F) \]

Where \( Y = \text{FEATURE OR NOT\_FEATURE} \)

- \( P = \text{Part of speech} \)
- \( W = \text{stemmed word} \)
- \( F = \text{frequency} \)

As mentioned above the probabilities \( P(Y|P) \), \( P(Y|W) \) and \( P(Y|F) \) are conditionally independent of each other. Initially the probability distribution is assumed to be uniform and then we use iterative EM algorithm to update the probabilities of each \( P(Y|P) \), \( P(Y|W) \) and \( P(Y|F) \). The initialization and the update steps of the EM algorithm are as follows.

**Initialization:** We start with uniform probabilities for \( P(y|p) = 0.5 \), \( P(y|w) = 0.5 \), \( P(y|f) = 0.5 \)

**UNKs:** We assume that there is one \( \text{UNKPOS} \), \( \text{UNKWORD} \) and \( \text{UNKFREQ} \) in Part of speech, words and frequencies respectively. The probabilities of a feature given these UNKs are also initialized with uniform probability distribution.

**Maximization step:**
For each sentence
For each word/pos/frequency
  calculate the posterior against each label
  \[ p(\text{feature}_i | \text{word}_j) = \frac{p(\text{feature}_i | \text{word}_j)}{\text{Sum(over all words)}_p (\text{feature/word}_k)} \]
  increment the count of \( \text{feature}_i \) with \( \text{word}_j \)
  \[ C(\text{feature}_i | \text{word}_j) += p(\text{feature}_i | \text{word}_j) \]
Renormalize counts
Iterate until convergence

**Updates for UNKS:**
At each update step an assumption is made that we can see an UNK being a feature with a probability of number of features per a sentence. This is not a blatant assumption, the rationale behind this assumption is that since there are some feature words in a sentence then we assume each sentence has an UNK (POS/WORD/FREQUENCY) which has a chance of being a feature at the same probability of finding a feature in a sentence.

**Classifier:**
Calculate the probability of a word being a feature or non feature based on the bayes formulation we mentioned above and the probability cutoffs are decided based on the training and test data. We found that due to the high probability mass assigned to UNKS we have to set a low probability cutoff (we found 0.15 for Canon review data) when testing on the same data (eg training on 90%
Canon data and testing on remaning 10%), but when we test it on different data sets we can increase the probability cutoff and the results are reported as below.

4.3.1 Results

When we trained on Canon data (90%) and tested on the remaining (10%) the values are as follows processed 11851 tokens with 2451 phrases; found: 2147 phrases; correct: 1938.

accuracy: 96.65%; precision: 90.27%; recall: 79.07%; FB1: 84.30
FEATURE: precision: 96.27%; recall: 84.33%; FB1: 89.90
NOT_FEATURE: precision: 84.26%; recall: 73.82%; FB1: 78.70

Training set: Canon, test set = Nikon and probability cutoff = 0.4

accuracy: 88.73%; precision: 54.26%; recall: 61.61%; FB1: 57.70
FEATURE: precision: 65.10%; recall: 73.93%; FB1: 69.24
NOT_FEATURE: precision: 43.43%; recall: 49.32%; FB1: 46.19

Training set: Canon reviews and Test set: Creative

processed 32349 tokens with 8344 phrases; found: 4259 phrases; correct: 2003.

accuracy: 81.54%; precision: 47.03%; recall: 24.01%; FB1: 31.79
FEATURE: precision: 64.21%; recall: 24.79%; FB1: 35.77
NOT_FEATURE: precision: 39.45%; recall: 23.47%; FB1: 29.43

Observations:

As expected the results are better when we test and train on the dataset for the same product. This also proves that the assumption that feature classification relies heavily on Part of speech, word and frequency is justified. The interesting case is when the training set is of same and different domain data. By domain we mean that camera reviews belong to camera domain. The second results indicate that when review data is for products from same domain the precision and F numbers are good.

Here are some examples

The pictures are razor sharp even in macro

Interestingly as we'll see later macro won't be classified as a feature by maxent. The reason here we are finding pictures and macro as features is because they both are nouns and the MLE probability values of both these features are the same. In maxent we'll see when we remove the independence assumption, context (words around) becomes and important factor and frequencies, part of speech can't be looked at as independent attributes.

it is light enough to carry around all day
its many features is easy

The above two examples can also be explained by the same idea of dominance of part of speech and word frequency to determine product features. Both light and features in the above case are nouns.

Examples where Bayes-EM classifier didn't work:

compact flash and one rechargeable battery

16 mb starter card

We see that since rechargeable is not a noun it's being classified as a NOT_FEATURE, but in data it's classified as feature.

In the second case the frequency of card is less and hence though it was feature our BayesEM classifier classified it as Not_feature.

In the second Bayes EM experiment we assumed that

\[ P(YWPF) = P(Y)*P(W|Y)*P(P|Y)*P(F|Y) \]

the diagram give for bayes formulation can be assumed where the arrows are reversed. Since the conditional independence assumption can't be assumed our results weren't good here and hence we are not reporting the results. We have submitted the code for it too.

Also we see that we could have not improved the results by stemming or part of speech family identification (explained under Maxent) because they over generalize the model. We could have used frequency bucketing and we did that in Maxent model explained in next section. Mainly each word here is treated independently in the Bayes EM model and hence there is not context information like did the noun follow an adjective or was there information like availability of number around the noun. This also explains additional motivation to do maxent where we factored the context information too.

We can clearly see that our independence assumption and adding no context information here are bottlenecks, and hence to overcome this we used Maxent classifier which's explained in next section.

4.4 MaxEnt Classifier

As the maximum entropy estimates gives us the least biased estimate possible on given information, it is maximally noncommittal with regard to missing information. Unlike the last EM model on Naïve Bayes where we assumed independence between the model's attributes (POS, Stemmed word and Frequency buckets) in the MaxEnt model we do not make any such assumptions. We used the
same derivative and objective function formulation as in Programming Assignment 3. Even the
getLogProbabilities has been implemented the same way. The point to note is that throughout this
section a MaxEnt feature should not be confused with the product feature.

Other than the code for extractFeatures, the major change to code has been the way the data has
been handled, a quick glance at the input data immediately shows that each line is of the format:

word <stem_word, frequency, partofspeech^ATTR:good, bad, inferior> GOLD_LABEL

stem_word: Stemmed word of the actual using porter stemmer.
frequency: How many times a particular word occurred in the data.
partofspeech: part of speech of the word.
ATTR: We observed that WordNet provides an option called -attr which shows if a
particular word has attributes or not. For example the word size has attributes - big, small and
similarly the word quality has attributes: inferior, superior etc. This information gives a useful clue in
classifying whether a particular word is a feature or not.
GOLD_LABEL: Whether the actual word is a feature or not. Possible values are FEATURE /
NOT_FEATURE.

What deserves more explanation from the project perspective is the MaxEnt features we used to
perform product review feature classification.

The actual word (Stem word), word_prelabel - Multi valued feature

From the data it can be seen that the actual words play a very important role in feature classification,
for example in the case of Canon reviews, words like quality, picture, speed etc have been mentioned
in several places. To avoid the ambiguity between classifications of similar words we used the
stemmed word in place of the actual word. We also store the word and the previous label combined
as a MaxEnt feature.

HASATTR - Binary attribute

This feature is turned on if the word has attributes returned from WordNet and 0 if there are no
attributes for a word according to WordNet.

Part Of Speech - First two characters - multi valued feature

Most of the features in the data come from the same part of speech family. What we mean by family
here is that - VBZ, VBD etc all fall under the VB verb family. Here are the probabilities of some
POS being termed as a product feature. We see that NNS and NN have high probabilities of coming
out as a feature and some probability mass has been assigned to CD, FW, VBG, VBD etc. Point to
note that the probabilities of a particular POS don't add up to one, but what the values mean is that
what is the probability that a word is POS (like N N) when it is labeled as FEATURE. We extract
the first two words of the POS because we care about the POS family rather than exact POS, this
also helps in avoiding the problem of model over fitting.
Frequency Bucket - multivalued feature

We used the exact frequency of a word as a feature. Unlike the EM model here, the results improved if we consider specific frequency instead of bucketizing them.

Word normalisation - multivalued feature

This MaxEnt features captures the structure in the actual word. Most of the words which are classified as product features have some alphabet/number arrangement in common. For example, we have product features like mp, g3. If a word is seen which is g4 then we should be able to identify this word as a feature and hence we introduced this normalization feature. Which converts a word, say Camera2 as

Xxxxx0 where every capital letter is replaced with X, small letters with x and numbers with 0.

Has_underscore - binary feature

We used this feature to see if there's an underscore in the label, this is the only format information that can be extracted from the label.

Length of the word - multivalued feature

This feature is useful to cluster words which have same length and are features. As normalization section and after capturing the structure of the word we need to identify how long it is. This helps in clustering words like mp, MP etc.

Next word POS - multivalued feature
After experimenting with features like - previous word, next word, previous word POS and next word POS, the most useful feature seems to be next word part of speech. We have words in our data like picture quality which are two words but are feature together. Both have the same POS noun as marked by the stanford parser and hence we need a way to capture words which occur together and are classified as features so next word POS feature was used.

4.4.1 Results

There are various ways of reporting results as we had 5 data sets. This was the most logical way according to us. First we'll train the data on 90% Canon and then test it on remaining 10%. The scores are as follows

processed 1189 tokens with 326 phrases; found: 320 phrases; correct: 316.
accuracy: 99.41%; precision: 98.75%; recall: 96.93%; FB1: 97.83
FEATURE: precision: 100.00%; recall: 97.83%; FB1: 98.90
NOT_FEATURE: precision: 98.37%; recall: 96.79%; FB1: 97.57

When trained on Canon and tested on Nikon reviews, the results are

accuracy: 91.79%; precision: 72.35%; recall: 70.66%; FB1: 71.49
FEATURE: precision: 77.41%; recall: 74.70%; FB1: 76.03
NOT_FEATURE: precision: 69.19%; recall: 68.09%; FB1: 68.64

When trained on Nokia and tested on Canon
processed 11852 tokens with 3054 phrases; found: 1735 phrases; correct: 1052.
accuracy: 85.11%; precision: 60.63%; recall: 34.45%; FB1: 43.93
FEATURE: precision: 79.25%; recall: 36.16%; FB1: 49.66
NOT_FEATURE: precision: 51.83%; recall: 33.32%; FB1: 40.56

When trained on Apex and tested on Canon
processed 11852 tokens with 3054 phrases; found: 1665 phrases; correct: 769.
accuracy: 82.12%; precision: 46.19%; recall: 25.18%; FB1: 32.59
FEATURE: precision: 63.17%; recall: 27.02%; FB1: 37.85
NOT_FEATURE: precision: 38.42%; recall: 23.96%; FB1: 29.51

From the above results we can infer that a good F number is obtained when we train on 90% data and test it on remaining 10% of the data. We used this to see tweak our features and come up with a good and minimal set of features. The results also show that the scores are high if the test is conducted on reviews of products from the same domain. We see that when the training set was Canon and the test set was Nikon we are getting a 77.41% precision on FEATURE classification and an F number of 71.5. Most of this can be attributed to commonality of language between these products. As expected the scores become bad when the products start differing by a great amount, this can be clearly seen from the drop of precision and F values. The least F value of 32.5 is obtained with the training set is Apex and test data is that of Canon.

Exact features: Here is an example of the features we could identify using MaxEnt in some sentences when trained on Nikon data and tested on Canon data.
Excellent **picture quality** for a camera I recently purchased.

Well made camera easy to **use**.

Very flexible and powerful **features**.

With the automatic **setting** I really have not taken a bad **picture**.

The bold words are the identified features.

**Examples where the MaxEnt classifier failed to identify features:**

- Picture are razor even in the **macro**.
- **Auto-focus** performs well but I love having the 12 optional **scene modes**.
- A 16 mb compact **flash** and a **battery**.

The features which were identified are in bold and the features in italics are the ones which our classifier didn't identify. The reasons could be that in the last sentence we don't have an adjective in front of battery like there is one before flash which boosted the score of flash, similarly the feature scene mode is preceded by optional. Auto-focus being a combination of two words separated by hyphen is not identified, similarly macro which doesn't have any adjective around it the classifier failed to identify them as product features.

**5. Implementation Details**

For data preprocessing and basic parsing we wrote a number of scripts in Python. For stemming the words in Python, we use the Porter Stemmer available in NLTK Python library [8]. To obtain the possible attribute qualifications for a word, we use WordNet [9]. To obtain POS tagged data, we used the Stanford Tagger as described in [6] and available at [7]. For the baseline system described in Section 4.1, we used Python scripting, and an open source APRIORI algorithm implementation [10]. We used the Naïve Bayesian classifier available in WEKA. The MaxEnt and EM classifiers were built upon the CS224N PA2 and PA3 assignments code. We used shell scripts to run multiple tests of our system with different train and test data.

**6. Conclusion & Future work**

- Our current focus has only been on finding features that were explicitly mentioned in the product reviews. An interesting extension would be to identify value/opinion words associated with a feature, and use these to infer if the user is specifying some feature.
Another possible avenue for improvement is incorporation of domain knowledge in the classification process. Say for example, in reviews about a Canon camera, we know that Olympus, Nikon, Powershot etc refer to proper nouns relevant to this domain, and hence improve the classification accuracy.

Also, performing anaphora resolution on the reviews, so as to better identify the entity being referred to in a particular sentence might help improve the system.

As part of future work we aim to train the classifiers on data from different domains so that our results improve when we test on different domain reviews.

We also plan to implement clustering were a set of features could be classified in a group. For example features like BR and bedroom are actually identical, we could look at the values a particular feature takes and identify the cluster for the features.

In maxent our context is restricted to the part of speech of one word ahead and one word before, adding more context could improve our results. We tried just adding next to next to words which didn’t provide much help. We have to see how adding more relevant context could improve our results.

From our models we conclude that maxent model outperforms Bayes-EM, Naïve Bayes and baseline models.

Language features like sentence structure, part of speech and frequency of words are very important features in identifying a feature. Context information helps as we see in the max ent model.

Training data from one domain can’t be directly used for extracting features from another domain.

Frequency bucketing, POS family, Stemming, word net attributes and maxent classifiers are most promising features and classifiers for product feature identification.

7. Team contributions
Nikhil: wrote scripts for data cleanup and input feature generation for all the classifiers. Worked on the Naïve
Praveen: Implemented the EM and MaxEnt classifiers.
Rahul: Implemented the initial approach and wrote scripts for testing and training on the different datasets.
Analysis of results and report were done togethr.

8. References