Sentiment Extraction of Earbud Amazon Reviews

1. Introduction

Amazon.com provides an abundant amount of reviews to aid buyers on their decision to purchase the product. However, it is very time consuming for buyers to read through all the reviews and filter them for only information relevant to them. Although Amazon attempts to ease the problem by ranking its reviews in order of helpfulness, the top reviews may only be considered a top review simply due to visibility. To solve this issue, we propose to automatically extract sentiments from the entire set of reviews.

Our basic approach is to first generate a word bank of relevant terms using LDA from the product descriptions and questions asked by purchasers about the product. Next, using SVM and SentiWordNet with Word Sense Disambiguation (WSD), annotate all sentences with a sentiment score and only use the non-neutral sentences in the n-gram generator to produce relevant adj-noun bigrams or trigrams. These terms will be filtered by the word bank produced by the LDA and ranked to produce a set of most relevant terms with their sentiment. The output will be terms that our algorithm decides to be the most relevant.

2. Problem Definition and Methods

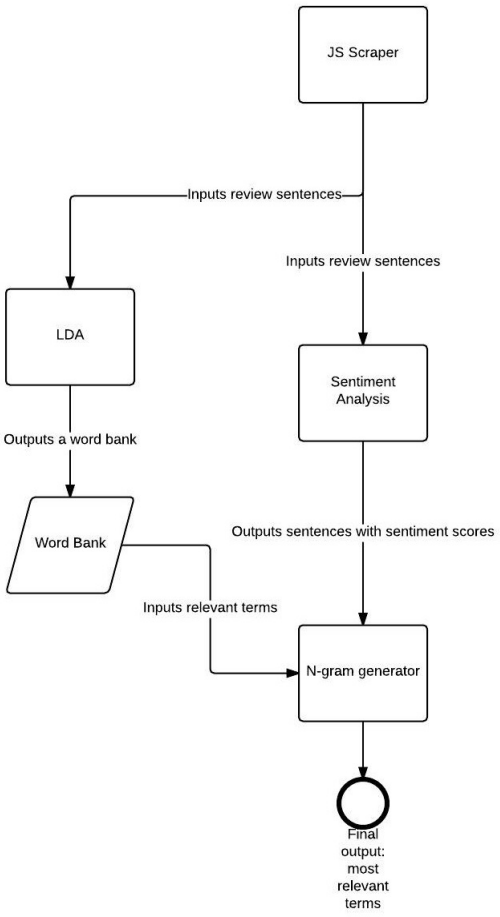
2.1. Task Definition

Our problem is to extract sentiment tagged aspects that will give a good representation of the text corpus it was generated from using the method described below. The main questions we want to answer are:

1. What are characteristics of products important to consumers?
2. What are important features reviewers focus on?
3. What makes review content irrelevant or less relevant?

It is an important problem, because accurate extraction of sentiment would save consumers lots of time in any e-commerce market place setting which relies on large numbers of product reviews to sway consumers to purchase a product. What we are proposing is an algorithm that can be applied to any product because it adapts to be context specific through analysis of prior information for each product. Furthermore, this can be applied to other areas involving text summarization, such as automated generation of catch phrases and key points for company ads

2.2. Algorithm and Methods



To answer the question of what product features and terms are relevant to a prospective consumer, we used product descriptions that we mined to form a corpus of documents which describe different earbuds/headphones products to generate contextual terms. These terms will refine the results of the n-

gram generator by using prior information provided for the products.

In previous works, it has been shown that LDA can be used as an effective unsupervised learning algorithm to cluster textual data into a series of highly probable topics that describe the corpus' contents. We chose LDA as it can run unsupervised and uses a generative model to determine what topics comprise the corpus of documents and how those topics can be described with certain text terms. In terms of the problem statement, this provides us with a basic understanding of the terms that product sellers believe is relevant to a consumer. We assume that the terms in the product description actually have some form of relevancy to the general products in the earbuds category. This is justified by the assumption that these product descriptions are formed based on the market research the product seller has done. This approach ignores the possibilities that certain terms are relevant to different classes of earbuds, and that the seller may have misinterpreted the interests of the consumer.

To use the LDA, we collected a large set of product descriptions manually mined from a variety of different product pages on Amazon and set each description to represent a document. We used the standard stopword filter to filter out words that are usually considered irrelevant when analyzing text. An unsupervised version of LDA was run, which takes in a set of documents, a stopword list and the number of topics that we want to extract. We utilized a k value equal to fifteen and ran the LDA in six different runs where each run ran LDA either fifty times or two-hundred fifty iterations. Using the results from the six runs we can corroborate that similar terms are extracted from each run implying that the terms are considered all interrelated and also relevant. This is necessary as LDA is a probabilistic model so different runs can extract different topics, however we believe that iterating on these independent runs can still lead to the extraction of similar tone/word topics. Through this, we hope to create a word bank of terms that are contextually related to the main product. We can use these terms as a resource when ranking the sentiments extracted, and filtering based on a relevancy threshold.

To extract relevant terms from the reviews, each sentence of the reviews was made into a POS-tagged parse tree and chunked on various grammars. Through multiple runs of chunking on different grammars, it was determined that chunking into bigrams of ¡adj¿*¡noun¿* and trigrams of ¡DT¿?¡adj¿*¡noun¿* were most accurate in extracting relevant terms. To offset the thousands of false positives the parse tree generated, the results of the parse tree were filtered by the word bank generated by the LDA. If the word existed in the LDA, it would be given heavier weight. At the end, only words with a weight above a certain threshold were considered relevant terms.

To give each sentence a sentiment score, we first train a bag of words with a combination of the movie review corpus from nltk and a portion of the annotated laptop reviews from Stanford SNAP library with all the unique words stemmed by using Porter Stemmer. With an interface to SentiWordNet using the NLTK WordNet classes, for each sentence, we split it into individual words and for each word in that sentence, we look it up in the SentiWordNet dictionary for its positive and negative sentiment scores. SentiWordNet is a document recourse which contains a list of English terms which have been attributed a score of positivity and negativity. By using the sentiment scores from SentiWordNet, we can make an educated prediction of the sentences expressed sentiment.

In order to obtain more accurate results, we also implemented WSD(Word Sense Disambiguation) and found the closest meaning of a given word w through the maximum path similarity of w and the other words in the sentence. The sentiment score of a sentence would be the summation of the weighted sentiment scores of the words inside that sentence. By weighted, we mean if this word appeared more times in the positive class, its positive sentiment score is multiplied by the log of its frequency in the positive class. If a word does not exist in the bag of words we obtained from training, it will simply be ignored.

To score each sentence with SVM, we use the annotated laptop reviews to create feature vectors representing word counts for each word encountered in the training data. The soft margin weight vector is then computed using SVMlight. This weight vector is used to assign a sentiment score to each sentence.

3. Experimental Evaluation

3.1. Methodology

1. Preprocessing stage:
   Using the scraper extract all user reviews and split by sentence to generate a list of sentences from all the reviews per product.

2. Processing stage

   2.1. Determining Important Features of each product
   This will be done with a combined static and dynamic technique.

2.1.1. Static Technique
Generate important aspects from product descriptions scraped from Amazon.com by using a LDA analysis. These words, along with the words sound, audio, comfort, bass, highs, mids, tones, durability, quality, blocking, noise, sharp, deep, and tangle will be considered important aspects automatically.

2.1.2. Dynamic Technique
Using a parse tree split each sentence into bigrams and trigrams with the grammar format of <adj><noun> to represent candidate product aspects of importance to reviewers. We can filter this list using the word bank generated from the LDA.

2.2. Using SVM, annotate each sentence with a sentiment score based on the previously created list of relevant features for the product. If a sentence has a score below a low threshold then it will be discarded.

3. Post Processing Stage The terms will be ranked so that only those with the highest scores will be considered. Sentences containing these terms are returned.

Evaluation: We believe the main problem can be solved through a composition of the above techniques to filter neutral sentiments, extract entities that may describe the product, rank the entities based on relevancy to product, followed by ranking the sentiments based on frequency of occurrence or aggregate sentiment towards a given entity.

The reasoning behind this is that the problem seeks to extract sentiments. A sentiment is defined as an authors opinion (positive or negative) about a given product feature (herein referred to as an aspect). We must first filter neutral sentiments which likely have factual information rather than opinions.

Next we seek to do a form of entity extraction using the n-gram generator so as to have a list of potential aspects that are being referenced in the sentence. These aspects could be potential targets of the sentences sentiment. This naively ignores the possibility of compound sentiment where there may be multiple sentiments with different targets. We justify this through the assumption most reviews usually spread their evaluation over a variety of areas and analyze each one individually.

Following the entity extraction which is based entirely on structural information in the sentence and POS tagging, we use contextual information to determine what is relevant to the product. Given the same exact review for a headphone product, an iPhone product, and a guitar product, the focus of what entities are relevant to the consumer will change. However using our LDA analysis we can ensure we only utilize the aspects consumers are most likely to be interested in.

To evaluate our results, we ran the different parts of our algorithm on laptop review data that was annotated with relevant terms per sentence and each of those terms was annotated with either positive, negative, or neutral sentiment. Since we wanted to get the overall sentiment of the sentence, each sentence was given a sentiment score based on majority vote of the sentiment of the terms extracted from it. The testing and training data are realistic, because it is also technology review data that focused on extraction of relevant terms and sentiment as a mean of text summarization.

Each part of the algorithm was benchmarked using annotated data to get accuracy of the individual parts. Our goal is to determine if we can answer the research questions using the components described for the overall goal of extracting relevant sentiments. We benchmarked them separately and evaluated them based on their performance at their individual task.

At the moment there is no effective way for us to evaluate the relevancy of the outputted sentiments, though theoretically it could be evaluated through the construction of an information retrieval problem where we try to see how relevant the top sentiments are to a corpus of consumer submitted questions on Amazon.

In order to determine if a review sentence is opinionated, we need to be able to apply sentiment analysis on each input sentence. We decided to experiment with two different approaches to see which one would have a better performance in terms of correctly classifying each sample.

One of the two approaches was using SentiWordNet and WSD to train and test a sentiment classifier on the annotated laptop reviews. Since only the opinionated sentences were relevant to our project and that SentiWordNet only supports a binary classification of either positive or negative, all the neutral data samples were ignored, which left us with 1220 annotated sentences. The sentences had aspect terms marked for each of them, and each aspect term had a sentiment. We simply took the majority vote on the sentiment of the overall sentence based on the sentiments of the aspect terms. The algorithm works as described in section 2.2, and we tested it with a four-fold cross validation

methodology in which each fold included 900 samples used as training data and the other 320 used as testing data. The accuracies were obtained by checking the classification produced by our algorithm against the assigned sentiment of the sentence from applying majority vote on its aspect terms sentiments. Below are the results:

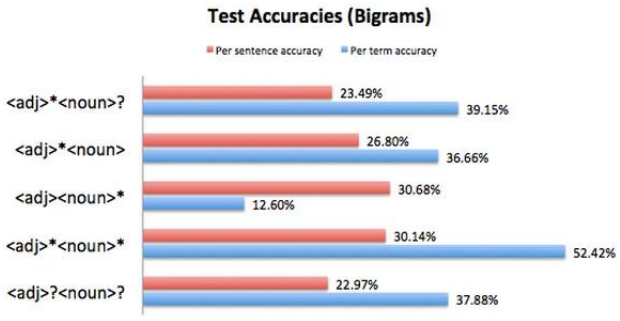
Fold Num.	Accuracy %	False Negs.	False Pos.
1	62.50	45	75
2	62.81	36	83
3	65.20	51	60
4	61.88	50	73

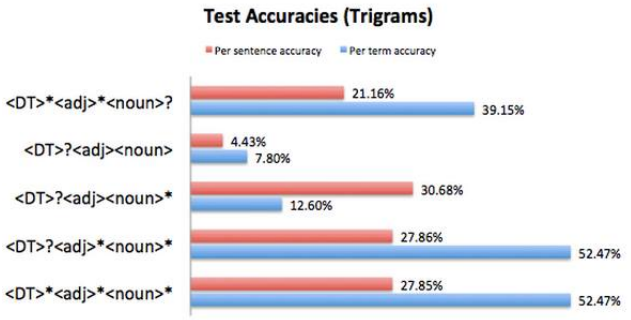
Average accuracy = 63.05%

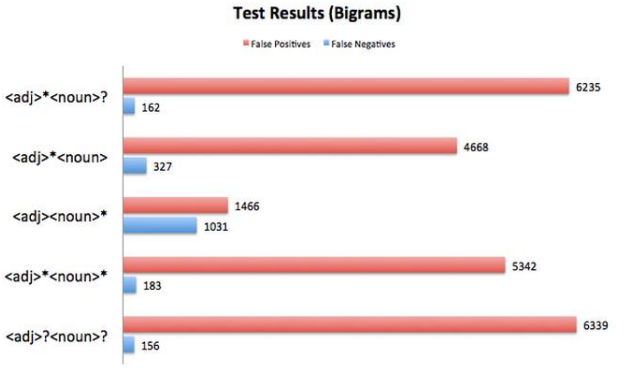
Our other approach was using SVM to train and test a sentiment classifier on the same annotated laptop reviews. To preserve consistency with our other method of evaluating sentiment, we also ignore all neutral data samples. The sentences were classified the same way as described above. We trained on a random selection of 70% of the training data, and used the remaining 30% as test data. The data was trained and tested with SVMlight on a soft-margin unbiased hyperplane. The results are as follows:

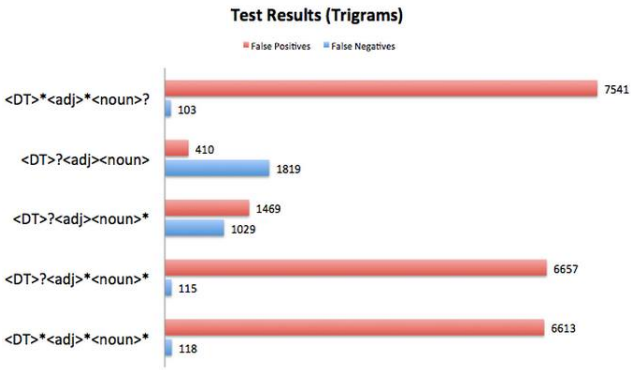
Accuracy %	Precision	Recall
74.33 %	70.33%	81.22%

The algorithm for extracting relevant terms from the parse tree was run multiple times using different grammars to chunk on in order to determine which grammar produced the most accurate results. In general, the tree chunked on adj ,noun bigrams or determiner,adj,noun trigrams. The results were checked against the annotated laptop review data we obtained. Majority of the review sentences in that dataset have aspect terms marked, so we checked the bigrams and trigrams produced by our n-gram generator to see if our results matched the annotated aspect terms of that sentence. The results are summarized below:









3.2. Results

3.3. Discussion

The high amount of false positives in term selection via parse tree chunking on bigrams and trigrams is due to the oversensitivity of the chunker. The parse tree will chunk on any phrase that fulfills the grammar, including terms such as father or mother, which are nouns but irrelevant to the reviews. This implies having contextual information is important and can benefit the n-gram generator. This also implies basic grammar chunkings may not be optimal and more specific grammars may need to be utilized.

The average accuracy for sentiment analysis with SentiWordNet and WSD was not satisfactory for multiple reasons. First, there was not sufficient or accurate training data since the annotated review sentences did not have a sentiment assigned to it as a whole. In-

stead, each sentence had aspect terms which each had a sentiment assigned, which was not very accurate in reflecting the sentiment of the entire sentence. This means neither of our training data or testing data was particularly reliable. Another issue that comes with this shortage of data means it is highly possible that when classifying a test data sample, relevant terms were ignored because they were not in the bag of words obtained from our training data, which would also negatively affect the overall accuracy. Finally, each word has multiple possible meanings depending on the context it is used, and a lot of the times, the meanings differ much from each other, which posed a major technical difficulty to our sentiment analysis method with WSD and SentiWordNet since it focused on individual words and lacks context. We did implement some level of WSD by assessing path similarities provided by WordNet among words that were in the same sentence, which did provide some context when choosing which meaning of a word the algorithm should pick, but our current approach to fully realizing WSD is far from perfect. Such unresolved issue was another cause of the unsatisfactory accuracy of this particular sentiment analysis algorithm.

Though the terms in LDA seemed to be relevant we need a better evaluation criteria to understand whether the results are valid. It does tell us that the terms found are relevant to the product, but we cannot measure how relevant they are or whether certain terms found are more relevant than others. One way we could theoretically test the effectiveness is by taking several topics extracted and an assortment of words using the topic distributions. We can then try to see whether the topic distributions are consistent with a human's perception of the terms. If we group several words from the same topic together and insert a foreign word, would a person be able to detect the intruder? This is a method that uses human cognition of the topic described through the association of the terms to determine whether the words are cohesively connected enough to be differentiated from a random word.

4. Related Work

Our method has two main benefits. One is that it can be reused for many different and related products and those results can be utilized to potentially improve overall accuracy. The sentiment analysis and n-gram generation for the most part can be reused as is. The only thing that needs to be generated product specific is the word bank of context specific terms, though even the process for that is exactly the same, merely the data taken needs to be different.

The other benefit is that we focus on what the users are interested in rather than just summarizing the raw sentiments contained in the text. Through focusing on what the consumer cares about we can target sentiments more relevant to their purchase decision. We also focus only on opinions and dont attempt to understand factual information that may be provided by the reviewer, which is a tradeoff as we focus on their opinions. Factual information extraction is more of an information retrieval problem, whereas our problem focuses on a summarized interpretation of the authors opinion towards a product feature.

5. Future Work

There are certainly many aspects we can work on to improve our results. For sentiment analysis, because of the limitation we had with only being able to work with binary classification, neutral samples were not included in the process, which could have been a great help in terms of more accurately determining which sentences were relevant. Learning neutrality can also potentially improve the classifiers learned distinction between positive and negative samples. One possible way of realizing this goal would be to combine pairwise learned classifiers like positive-negative, positive-neutral and negative-neutral, respectively. Another shortcoming we had with sentiment analysis of sentences using either of our two approaches was that none of them took into account the sentence structure as a whole, which could have a significant impact on what sentiment a sentence is conveying alongside with which words occurred in that sentence. With this in mind, investigating the possibility of using a decision-tree-based algorithm or random forest to include sentence structure in the classification process would be another promising improvement we could have with sentiment analysis. Our issue, which was mentioned earlier in the paper as well, with not having sufficiently accurate training or testing data needs work as well. Therefore, one other area we could put more effort into would simply be collecting more well-annotated data that provides reliable reference in sentiment classification.

6. Conclusion

7. Resources

Python libraries: Sci-Kit Learn, NLTK, Numpy, Scipy

Data for benchmarking and testing accuracy

- Web scraper written in Node.js to parse relevant data (reviews, product descriptions, etc)

- Annotated data from SemEval2014-Task4 on aspect feature based sentiment analysis
  http://alt.qcri.org/semeval2014/task4/index.php?id=data-and-tools

- Amazon product review data mined by Stanford
  http://snap.stanford.edu/data/web-Amazon-links.html

Readings on Parsing & Sentiment Analysis:

- Stanford paper on parsing & sentiment analysis:
  http://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf

- Multi-aspect sentiment analysis
  http://www.cs.cornell.edu/home/cardie/papers/masa-sentire-2011.pdf

- Catching the Drift: Probabilistic Content Models, with Applications to Generation and Summarization
  http://www.aclweb.org/anthology/N04-1015

Code citations

- Sentiment classifier prototype by Kathuria Pulkit
  https://pypi.python.org/pypi/sentiment_classifier

- SVM-Light by Thorsten Joachims
  http://svmlight.joachims.org/

- LDA in JS code by David Mimno
  http://mimno.infosci.cornell.edu/jsLDA/index.html