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, but it is very time consuming to read and filter through all the reviews for relevant information. To solve this issue, we propose to automatically extract sentiments from the entire set of reviews and aggregate it in a concise summary.

Our basic approach is to generate a word bank of relevant terms using LDA from the product descriptions and questions asked by purchasers on the product. Next, using SVM and SentiWordNet with Word Sense Disambiguation (WSD), annotated 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 the sentences these terms were produced from.

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:

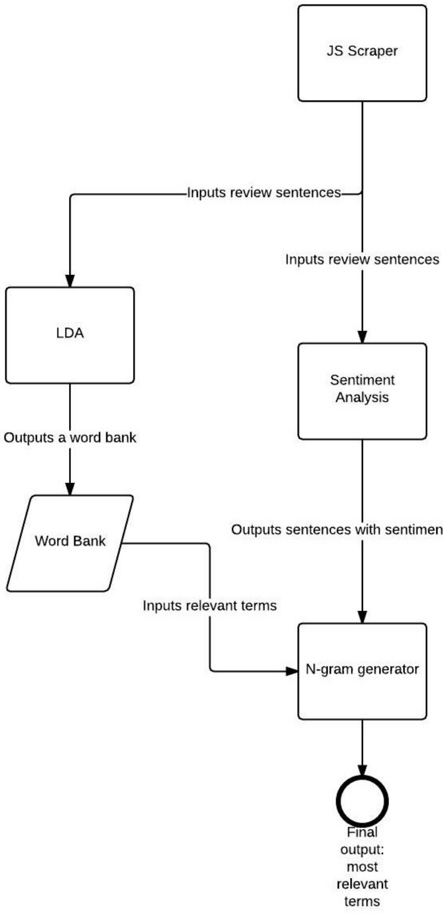
1. What are characteristics of products important to consumers?

2. What are important features reviewers focus on?

3. What makes review content irrelevant or less relevant?

It is an important problem, because if our algorithm accurately extracts the sentiment, we will be saving consumers lots of time on not just Amazon.com but other large marketplaces. Furthermore, this can be applied to other areas involving text summarization, such as automated generation of catchphrases and key points for company ads.

2.2. Algorithm and Methods



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. In order to obtain more accurate results, we also used WSD with finding the closest meaning of that particular word by determining the similarity that word has with the other words in the same sentence. The sentiment score of a sentence would be the summation of the sentiment scores of the words inside that sentence. If a word does not exist in the bag of words we obtained from training, it will simply be ignored.

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 TFIDF weighting scheme from NLTK. 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 stopword filters and other grammatical based rules to get rid of false positives. We can create an inverted index of these terms and the sentences they occurred in. We can also easily use the list of sentences to find the frequency of aspects in the review corpus.

   2.2. Using SVM or Naive Bayes 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.

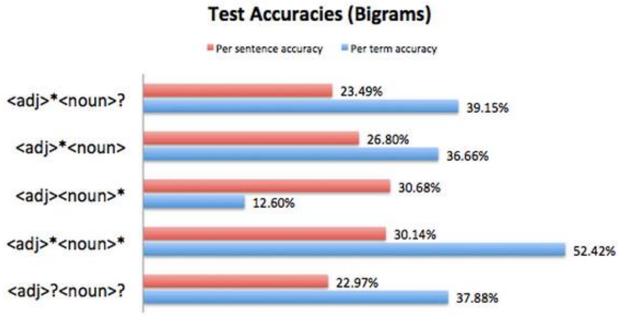
3. Post-processing stage: Aggregation

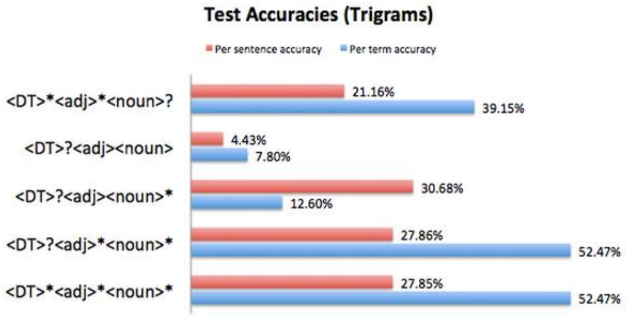
   3.1. We need to summarize the aspect information we have and present sentences from the corpus that best represent the corpus. One way to do this is to rank the aspects based on frequency and then calculate an averaged sentiment score, which can be used to determine which sentiment polarity the majority of the reviews have with respect to that aspect. We can then take a sentence to represent this aspect. A simple scheme could be to find the sentence with the greatest magnitude sentiment score with the same polarity as the aspect average score.

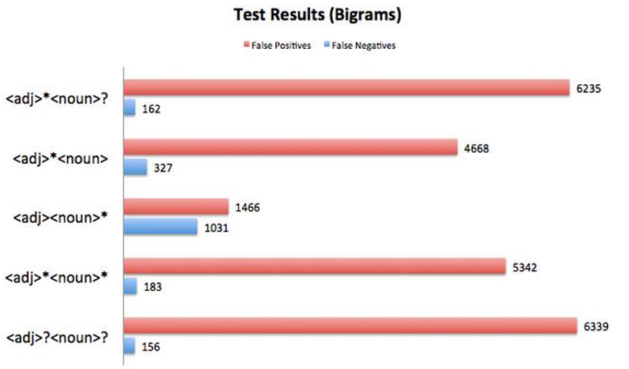
Evaluation: Compare the results with annotated libraries of training and testing data from a project on reviews to find our accuracy on tagging aspect features and sentiment intensity/polarity. After that we will have to do manual surveying from people of the different summarization techniques that we can use.

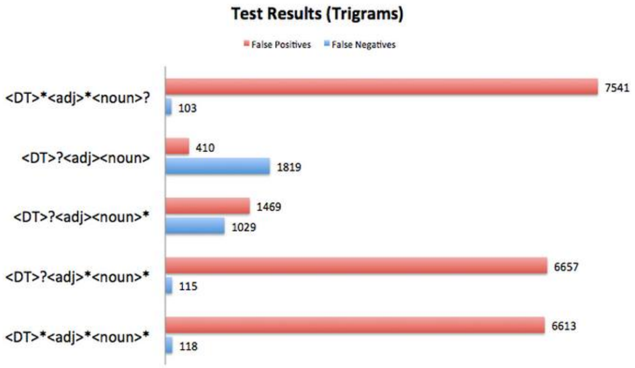
To evaluate our results, we ran the different parts of our algorithm on laptop review data that was annotated with relevant terms per sentence and each of those terms was annotated with either positive, negative, or neutral sentiment. Since we wanted to get the overall sentiment of the sentence, each 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 run separately on the data to get accuracy of the parts. We assume that if the parts are accurate, the overall goal of extracting relevant terms with sentiment will be accurate as well. The preliminary algorithm was (Chetan write stuffs). 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 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. Instead, 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 would 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, even though we implemented some level of WSD by assessing path similarities among words in the same sentence and tried to deal with determining which meaning to take for one particular word since a word can have different meaning given its context, but the current approach to realizing WSD is far from perfect, which was another major cause of somewhat low accuracy.

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

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

- 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