CS4780 Project Proposal

1. Size of Team

Size of team is four people.

2. Motivation

Before making a purchase on Amazon.com, buyers typically read reviews of the product to determine if it will meet their requirements. Although reviews are a great resource, it is very time consuming to read a large volume of reviews to find the information that is relevant to them. Although Amazon attempts to shorten this process by ranking its reviews in order of helpfulness, the top reviews may still not be enough for a buyer to make an informed decision. The top reviews may not address all the product features that the buyer is interested in, may be too old to still be reliable, or may still be considered a top review simply because of visibility.

Furthermore, some buyers may go beyond the top reviews to get a majority opinion instead of relying on a few experiences. To do this, they then have to filter through a large number of irrelevant information and still only get a holistic view of what might be the majority opinion.

To solve this issue, we propose to automatically extract relevant information from the entire set of reviews and present it to the potential buyer in a format that makes it easier to digest and to compare different products.

3. Statement of Problem and Key Questions

Using user reviews, help consumers easily and accurately decide whether a product meets their requirements.

Key Questions:

1. What are the characteristics of the product that reviewers find important?

2. How do reviewers rate these features?

4. General Approach

We will restrict our focus to one type of product (headphones) to make evaluation easier. We see 3 main stages to this project:

1. Preprocessing stage: Extract relevant product features

   1.1. In order to determine if a review contains relevant features, we will extract the product specifications from the product description using a web scraper. Using TFIDF from NLTK we will generate a list of words that have high frequency. These, in addition to the words sound, audio, comfort, base, highs, mids, tones, durability, quality, blocking, noise, sharp, deep, and tangle will be considered important aspects.

   1.2. 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. Using SVM or Naive Bayes annotate each sentence with a sentiment score based on the previously created list of relevant features for the product.

   2.2. 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. Using a hashmap store the phrase with the sentence it came from. Find the frequency of all these phrases in the entire set of sentences from reviews.

3. Post-processing stage: Aggregation

   3.1. A simple approach would average the sentiment scores from each sentiment the aspect was tagged in giving an output sentiment score for the aspect. We can then output the sentence with highest sentiment score tagged with that aspect with same sign as output sentiment score for the aspect.

Evaluation: Compare the results to the hand annotated results of the Google research document.

5. Progress

6. Resources

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

Resources to extract data from Amazon.com:

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

7. Schedule

By November 11th, the due date of the progress report, complete the following:

- extract data from Amazon.com either via API or self-implemented scraper; alternatively, use the Stanford dataset
- parse data into JSON or csv
- filter out irrelevant reviews and weight constructive reviews more
- extract sentiments from review data
- create a dashboard GUI to allow users to evaluate and understand data output

By December 4th, the due date of the poster, complete:

- aggregate sentiments using the dashboard to deal with conflicting sentiments for same feature by aggregating and displaying them.

  For instance: If 4 people say they like the texture of a phone case, but 6 people say it slips around too much, display 40% like texture, 60% dislike texture.
- rank sentiments in terms of relevance to consumer requirements (can use combination of either product description, weight from all the reviews that expressed the sentiment, or an alternative method to relate the sentiment back to corpus of reviews and use the frequency to determine rank)

Final report due: December 10th