ReviewsHub

CS 410 Final Project Report

May 2014

Xiaodan Zhang
Haoyan Cai
Zheng Kang

Abstract

ReviewsHub is an integrated search and comparison system that allows users to focus on a specific feature of a product. Based on thousands of Best Buy reviews, ReviewsHub offers a cross-cut overview of comments that discuss a certain feature, as well as a dynamic visualization that provides insight into how people compare similar products.

Introduction

Background

According to *Commerce Times*, 85 percent of all Internet users shop online. Attractions of online shopping include shopping during off-hours, a wider range of selection, cheaper price for better products, not having to drive to the store, and the guarantee that no one is judging.[1] Compared to shopping in stores, however, online shoppers don't have physical access to the product and rely heavily on customer reviews made by other customers. While these reviews are a great help, online shoppers' focuses and requirements vary from one another. User reviews play an important role in their decision-making process because these reviews provide many useful answers to the questions prospective buyers would want to ask. As a result, we wish to utilize information retrieval techniques to transform raw customer reviews into more useful feedbacks for online shoppers, which will significantly improve their online shopping experiences.

Goals and Challenges

ReviewsHub takes as input a collection of Best Buy reviews and two queries given by the user.

In `Rankr`, the two queries are a feature name and a product name, and ReviewsHub outputs a list of reviews sorted by relevance. It also gives a 4-sentence summarization for long reviews.

In `Visualizr`, the two queries are a comparison pattern (e.g. "better than") and a product name; ReviewsHub then scans through all reviews pertaining to the product specified and generates a graph detailing how people used that comparison when discussing the product (e.g. "X is way better than Y").

The biggest challenge we face in this project is how to filter the reviews in a semantics-aware manner. We also aim to treat the collection of reviews as a whole and try to extract information not seen in any single review.

Work Summarization

We built ReviewsHub as a Django web app. We used a variety of tools and libraries to help build the desired functionalities and tweaked the tools when necessary. As described above, ReviewsHub contains two major components: `Rankr` and `Visualizr`.

Related Work

Our tool enables users to search in product reviews with specific key words. Moreover, we visualize comparison patterns in our search engine. To the best of our knowledge, there are no other systems that provide the same functionalities. However, it is worthwhile looking into a few similar systems.

Amazon Most Helpful Customer Reviews

Amazon is one of the richest resources for online customer reviews. Buyers need to have a verified purchase in order to leave a review for an Amazon product. One would give an overall rating for the purchase and leave a comment that is longer than a length threshold. Once a review is published, other shoppers may comment or determine whether this is review is helpful. Then the Most Helpful Customer Reviews are returned.

Consumersearch.com

Consumersearch.com analyzes all its reviews to return top choices given users' inquiries. It helps users find answers about what products are top-rated or best bets in their class. According to their website, "Our website is partly a search tool and partly a consolidator of wisdom and analysis." [3]

These systems are good examples of how online commerce systems drive their business needs by utilizing and analyzing online review data. Our tool is a good supplement in that we try to facilitate users with the abilities to retrieve reviews tailored by their individual focuses.

Methods

We obtained thousands of Best Buy reviews via Best Buy's Review API[2] and preprocessed them offline for storage in a database. We decided to download and preprocess reviews because we believe these reviews are fairly representative and customer feedback regarding a certain product is unlikely to change dramatically overnight.

Search Engine

ReviewsHub makes use of Whoosh, a search backend purely implemented in python, and Haystack, a Django plugin that bridges the gap between our web app and Whoosh. In the file `search_indexes.py`, we defined what we want indexed so that Whoosh can correctly generate forward and inverted indexes. By default, Whoosh ranks the documents using BM25 which we left unmodified.

Visualizer

ReviewsHub utilizes a python web mining module, *Pattern*.[4] Specifically, this Visualizer function uses the pattern.graph module that provides a graph data structure that represents relations between nodes. Graphs are exported as HTML <canvas> animations and more central nodes that indicate having more incoming traffic are colored in blue. In the file `visualization.py`, We defined a pattern, "`p = '{NP} (VP) ' + compare_phrase + ' {NP}'`" in `compare_visualization()` so that the exported animated graph displays a directed graph of all reviews containing the specific pattern, and shows how one product is compared with other products mentioned in the reviews. This module is free, well-document and easy to use.

Summarizer

In a well maintained online review community, many users post reviews that are very long but contained much more detailed information making it quite hard to digest valuable information from these long reviews. This is where a summarizer should step in. A comprehensive summarization tool should utilize a variety of techniques including natural language processing, text mining and search engines. Considering the time frame of our project, we are only using a very simple algorithm[5] based on Classifier4J[6], an automatic text summarization tool. We summarize a review string when it is longer than a pre-specified length threshold.

First we remove stop words according to NLTK's English stop word lists. Then, we determine the most frequent words in this long review string. Sentences that contain most frequent words would be output according to the original order of sentences in the review string. In our demo, we restrict review string length to be 1500 and output 4 sentences as a summary.

Usage and Evaluation

ReviewsHub website has four search boxes on the homepage: two boxes under the `Rankr` function and two boxes under the `Visualizr` function. The following shows how to use/evaluate the two ReviewsHub functions:
For the `Rankr` function,
1) Type key words/features about a product into the first search box. For example, "battery life".

2) Type the product name into the second search box. For example, “galaxy s iii”.
3) Click on the “Search” button.

Then, the user will be directed to the “Search Results” page with the key words highlighted and all reviews well ranked according to their helpfulness. The user will also see a sentence like “You searched: battery life in product: Galaxy S III” indicating which features of a specific product he or she looks for. On top of the page below the “Search Results” title, there is a hyperlink to the home page. Additionally, for long reviews that are more than 1500 characters, ReviewHub’s Summarizer function helps summarize them into four-sentence reviews, which gives users an easier reading experience.

Below is a snapshot of the search functions. It can be seen that shorter reviews that contain searched terms are ranked higher than longer ones, which is because BM 25 penalizes long documents.

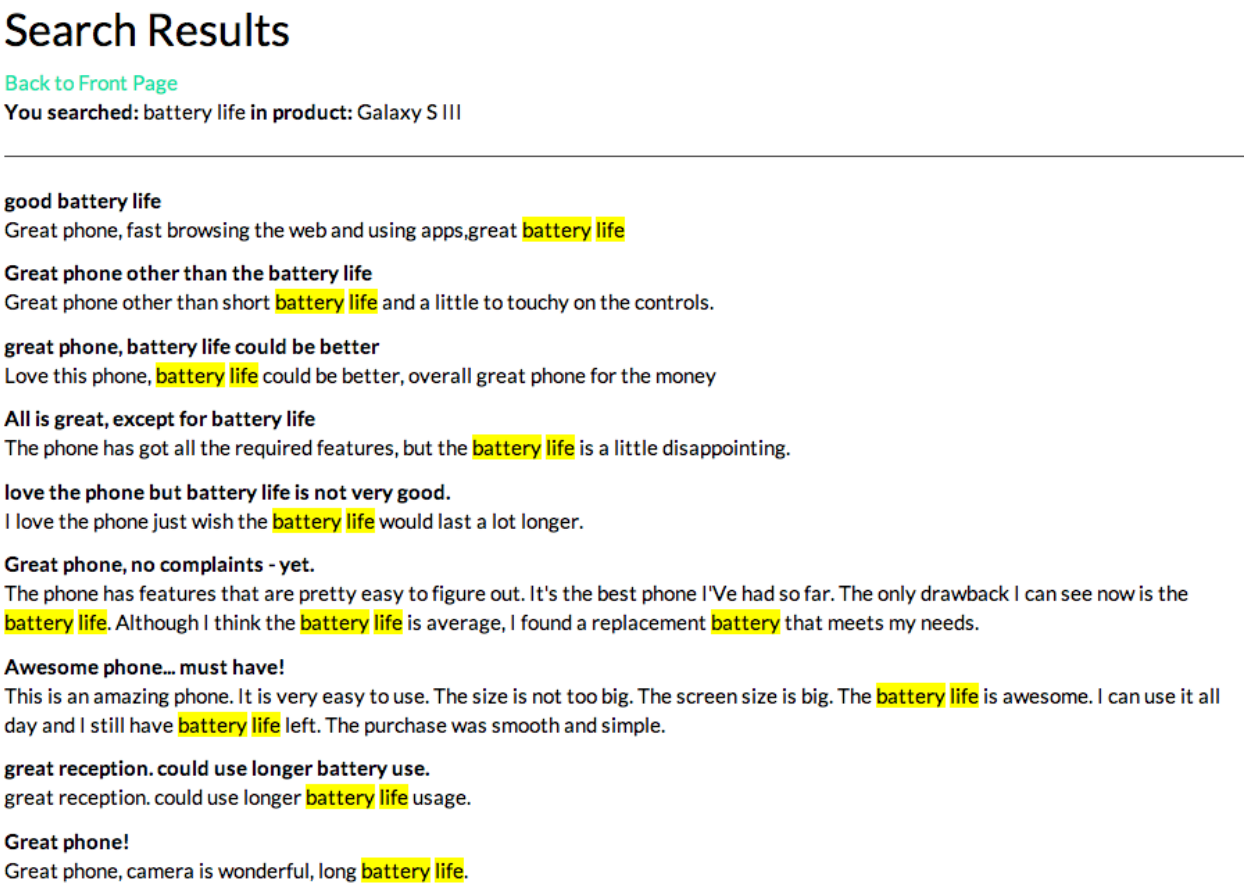

Figure 1. A screenshot of search results corresponding to the “battery life” feature for “Galaxy S III” product

Also, the summarizer function outputs usually starts with the first sentence of the original string. This is in agreement with general writing of habits of writing a general sentence at the beginning and elaborate in the rest of the documents.

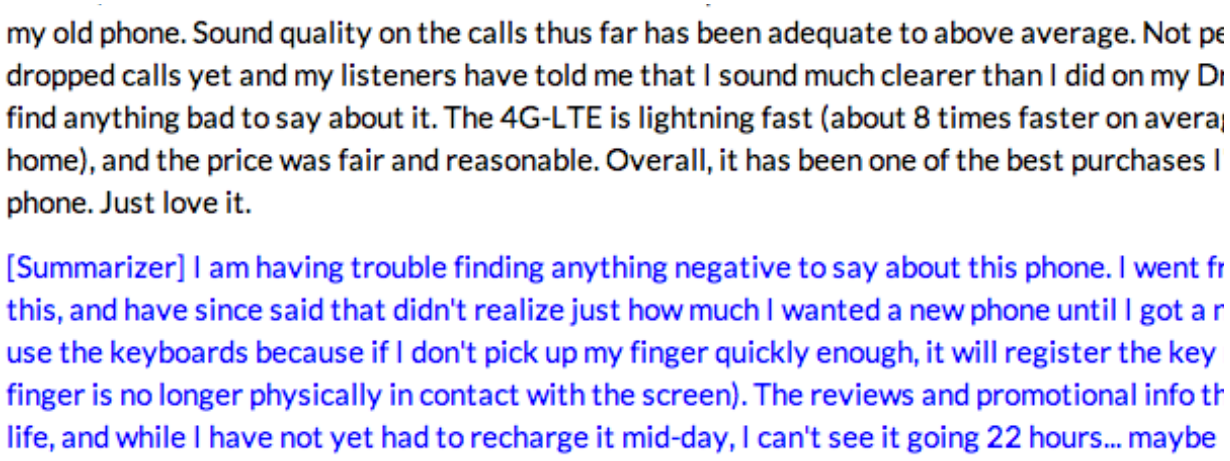

Figure 2. A screenshot of a summarized four-sentence review of a lengthy review

For the `Visualizr` function,
1) Type a comparison phrase about a product into the first search box. For example, “better than”.
2) Type the product name into the second search box. For example, “galaxy s iii”.
3) Click on the “Search” button.

Then, a new page containing an animated directed graph is opened. The graph shows relationship between nodes/terms. For example, the user may see that a node named “it” points to a node named “my old blackberry torch”. This can be interpreted as: the “Galaxy S III” is better than “my old blackberry torch”. The user may also see a node named “apps” points to a node named “the iphone”. This can be interpreted as: “apps” on “Galaxy S III” are better than those of “the iphone”. Moreover, the node named “the iphone” is colored in dark blue, indicating that many reviews contain this kind of comparison pattern and talk about the “Samsung Galaxy S III” is “better than” “the iphone”. However, if the comparison phrase is not found in all reviews, the user will be directed to a “Not Found” page. It shows a “Sorry, NO Patterns Found” warning message and a hyperlink to close the current tab.

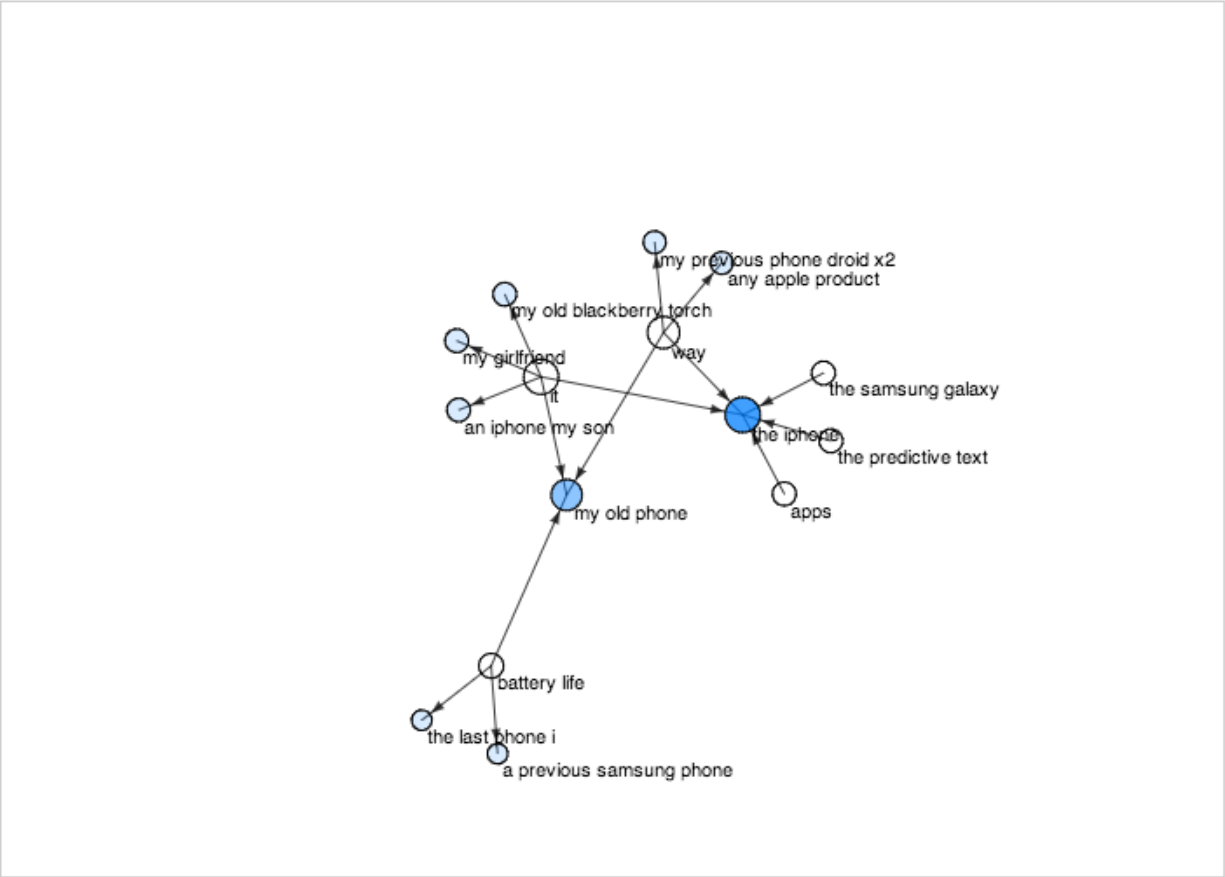

Figure 3. A screenshot of "better than" comparison visualization plot for "Samsung Galaxy S III" product.

In general, ReviewsHub website is user-friendly. However, it still has some deficiency: users get too many reviews returned; the result page looks too simple and not attractive at all; it would be better if the data visualization result page would be shown in a directed page instead of in a new tab; for long reviews, a button could be added at the end of them, so that the user could choose whether to see the summarizer result or not.

Conclusions and Future Work

We completed tasks in our proposal. We built ReviewsHub, a system to help users find helpful reviews specific to their concerns. We have three working functions: Search Engine, Summarization and Visualization with knowledge we learnt in the CS 410 course. We list limitations and future work as follows:

1. Search Engine:

Our search engine uses BM25 to determine the relevance and ranking of reviews given user inputs. BM 25 penalizes long reviews, directly resulting that most top ranked reviews are short ones that contain high frequencies of searched terms. However these top ranked ones are not necessarily most interesting or helpful to our users. As a matter of fact, people favor long reviews which contain several aspect of the product and with more details. This implies that though a review search engine is inherently a search engine, it has more complications. The ranking function needs not only consider traditional term and document frequencies but also take into account of users' desire to read detailed information. We can either change the penalizing parameters of the ranking function or we extract other features that indicate the amount of information in the reviews so as to give a better ranking score.

2. Summarizer:

The summarizer we are using is primarily based on frequent words. It would not work well when there are more than one topics in the review. Because the topic that has lower term frequencies may be filtered out. To address this issue, we would like to apply some techniques similar to the idea of Maximal Marginal Relevance Ranking to remove the redundancies and meanwhile provide richer information.

Appendix

Individual Contributions

Haoyan Cai:
1) Worked on the implementation of ReviewHub's Summarizer function

Xiaodan Zhang:
1) Worked on the implementation of ReviewHub's Compare Visualization function
2) Modified the front-end user interface

Zheng Kang:
1) Constructed the back-end part using Django framework
2) Worked on the implementation of ReviewHub's Search Engine function

References

[1] http://www.ask.com/question/what-percentage-of-people-shop-online
[2] Best Buy Reviews API, BBYOPEN https://bbyopen.com/documentation/reviews-api
[3] http://www.consumersearch.com/faqs
[4] Pattern | CLiPS, *Computational Linguistics & Psycholinguistics Research Center,* http://www.clips.ua.ac.be/pattern
[5] Summarize, *A python library for simple text summarization,* https://github.com/thavelick/summarize
[6] Classifier4J, *a Java library designed to do text classification,* http://classifier4j.sourceforge.net/