Review Opinion Diversification

Shreyansh Singh
Computer Science and Engineering
Indian Institute of Technology
Varanasi, India 221005
shreyansh.singh.cse16@iitbhu.ac.in

Abstract

This is our system as a submission to the task "Review Opinion Diversification".

1 Introduction

The problem of the shared task is "Review Opinion Diversification". This system aims to produce opinion matrices for the reviews of each product. In this system, we study the problem of generating feature opinions of customer reviews of products sold online. We are given a set of customer reviews of a particular product, the system performs three subtasks: (1) identifying features of the product that customers have expressed their opinions on (called product features); (2) for each feature, identifying their orientation i.e positive, negative or neutral opinions; and (3) producing an opinion matrix for each product with columns as the opinions and the rows as reviews.

2 Related Work

My system is heavily based on Bing Liu's paper (Hu and Liu, 2004b,a). The papers aim to solve the problem of opinion features extraction and orientation identification of product reviews. Bing Liu has done a lot of work in this field. His paper (Bing Liu, 2005) provides a great insight on the process of categorising opinions as positive or negative. His paper (Bing Liu, 2008) is a great work to identify the implicit features in a review as well.

3 Proposed Technique

3.1 Pre-Proceesing of Dataset

This part involves cleaning the dataset through stopword removal, lemmatizing. Also since we are dealing with customer reviews there are many misspelled words and different forms for the same word. For example - "reorder" and "re-order" actually refer to the same word. So we use a word autocorrector to correct the misspelled words and get a common form of all such words. So now all occurences of "re-order" are replaced by "reorder", i.e a common form.

3.2 Noun Phrase Extraction

Before discussing about Noun Phrase extraction, we first give some example sentences from some reviews to describe what kinds of reviews we are handling. Our system aims to find what people like and dislike about a given product. Therefore finding out the product features that people talk about is an important step. However, due to the difficulty of natural language understanding, some types of sentences are hard to deal with. Let us see some easy and hard sentences from the reviews of a Lawn Mower -

"The machine works really well."
"A very efficient mower."

In the first sentence, the user is satisfied with the overall working of the machine. Here, machine is the feature that the user talks about. Similarly, the second sentence talks about the efficiency of the machine, which is the feature here in this opinion. While the features of these two sentences are explicitly mentioned in the sentences, some features are implicit and hard to find. For example,

"Bulky, so not that usable and friendly."

Here, the customer is talking about the size of the lawn mower, but the word "size" is not explicitly mentioned in the sentence. To find such implicit features, semantic understanding is needed, which requires more sophisticated techniques. However, implicit features occur much

less frequent than explicit ones. Thus in this system, we focus on finding features that appear explicitly as nouns or noun phrases in the reviews. To identify nouns/noun phrases from the reviews, we use the part-of-speech tagging.
In this system, we use NLTK's POS tagger to do part-of-speech tagging of each word (whether the word is a noun, verb, adjective, etc). For extracting Noun Phrases, we use NLTK Chunker. We provide a regex pattern as an argument to the RegEx parser to chunk out the noun pharses.
We now make a transaction file for the generation of frequent features in the next step. In this file, each line contains words from a sentence, which includes only preprocessed nouns/noun phrases of the sentence. Here, nouns/noun phrases are used because other components of a sentence are unlikely to be product features.

3.3 Frequent Features Generation

This step is to find features that people are most interested in and mention the most. In order to do this, we use Association Rule Mining (Agrawal and Srikant, 1994) to find all frequent itemsets. In our context, an itemset is a set of words or a phrase that occurs together.
Association rule mining is stated as follows: Let $I=\{i_1, \ldots, i_n\}$ be a set of items, and D be a set of transactions (the dataset). Each transaction consists of a subset of items in I. An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \phi$. The rule $X \rightarrow Y$ holds in D with confidence c if $c\%$ of transactions in D that support X also support Y. The rule has support s in D if $s\%$ of transactions in D contain $X \cup Y$. The problem of mining association rules is to generate all association rules in D that have support and confidence greater than the userspecified minimum support and minimum confidence.
Mining frequent occurring phrases: Here we use the transaction file generated in the last section. To mine the frequently occuring phrases we run the association rule miner, CBA (Bing Liu, 1998), which is based on the Apriori Algorithm (Agrawal and Srikant, 1994).It finds all frequent itemsets in the transaction set. Each resulting frequent itemset is a possible feature. In our system, we define an itemset as frequent if it appears in more than 0.5% (minimum support) of the review sentences and has confidence greater than 50%.
We use a JAVA implementation of CBA (Coenen). First, we assign each word in our transaction file a specific number, and create a new file consisting with those words replaced by their respctive numbers. This file acts as the input for the CBA implementation. We use the "Frequent Sets" part of the output of the software as our required output. Since the output is also in the form of numbers, we convert the numbers back to words. The list obtained as the output of this software are the frequently occuring phrases (although it is not perfect).
The generated frequent itemsets, are referred to as candidate *frequent features* in this paper and are stored for further processing.

3.4 Feature Pruning

Of all the frequent features generated by association mining, some are not useful and genuine. There are also some uninteresting and redundant ones. Feature pruning aims to remove these incorrectly mined features. We have implemented two types of pruning as given below -
Compactness Pruning: This method aims to remove those features that have atleast two words and are likely to be meaningless.
In association rule mining the algorithm does not consider the position of an item (or word) in a transaction (or a sentence). However, in a natural language sentence, words that appear together and in a specific order are more likely to be meaningful phrases. Therefore, some of the frequent feature phrases generated by association mining may not be genuine features. The idea of compactness pruning is to prune those candidate features whose words do not appear together. We use distances among the words in a candidate feature phrase (itemset) to do the pruning.
We define a *compact phrase* as -

- Let f be a frequent feature phrase and f contains n words. Assume that a sentence s contains f and the sequence of the words in f that appear in s is: w_1, w_2,..., w_n. If the word distance in s between any two adjacent words (w_i and w_{i+1}) in the above sequence is no greater than 3, then we say f is compact in s.

- If f occurs in m sentences in the review database, and it is compact in at least 2 of the m sentences, then we call f a compact feature phrase.

For a feature phrase and a sentence that contains

the phrase, we look at the position information of every word of the phrase and check whether it is compact in the sentence. If we could not find two compact sentences in the review database, we prune the feature phrase.

Redundancy Pruning: This step mainly focuses on removing redundant features containing single words.

We define *p-support (pure support)* as -

- *p-support* of feature *ftr* is the number of sentences that *ftr* appears in as a noun or noun phrase, and these sentences must contain no feature phrase that is a superset of *ftr*.

For example, if the feature *automatic* has a support 10. It is a subset of the feature phrases *automatic mode* and *automatic device*, and suppose these features have support 4 and 3 respectively. Then the p-support of *automatic* is 3. Note that we require the feature to appear as a noun or noun phrase as we do not want to find adjectives or adverbs as features.
So we used the minimum p-support of the single word features to prune those redundant features. If a feature has a p-support lower than the minimum p-support (in our system, we set it to 2) and the feature is a subset of another feature phrase (which suggests that the feature alone may not be interesting), it is pruned. In our system, for the above example all the three features, *automatic*, *automatic mode* and *automatic device* will be taken as relevant features.

3.5 Opinion Words Extraction

Opinion words are words that people use to express a positive or negative opinion. Opinion extraction is done in the following manner -

- We go through all the sentences in the reviews. For each sentence, if it contains any frequent feature, then we extract the closest *adjective* in the neighbourhood of that feature. A nearby adjective refers to the adjacent adjective that modifies the noun/noun phrase that is a frequent feature.

In this way, we buid up an opinion word list which is used in the next section.

3.6 Orientation Identification for Opinion Words

For each opinion word we found for the different features, we need to identify and give its semantic orientation. The semantic orientation of a word indicates the direction that the word deviates from the norm for its semantic group. Words that encode a desirable state (e.g., beautiful, good) have a positive orientation, while words that represent undesirable states have a negative orientation (e.g., bad, sad). While orientations apply to many adjectives, there are also those adjectives that have no orientation (e.g., internal, manual) (Vasileios Hatzivassiloglou, 2000). In this system we find and assign positive and negative orientations and for the words for which could not get a proper orientation we assign them as neutral.
In this system we use the adjective synonym set and antonym set in WordNet (Miller and Beckwith, 1990) to predict the semantic orientations of adjectives. Specifically, we use implementation of WordNet in NLTK. The synonyms were found from the synsets of the word and antonyms were included from the antonyms list for each lemma of the synsets of the word.
In general, adjectives share the same orientation as their synonyms and opposite orientations as their antonyms. Using this idea, we first make a seed list of adjectives whose orientations are known. To have a reasonably broad range of adjectives, we first manually come up a set of very common adjectives (in our system, we have used 30) as the seed list, e.g. positive adjectives: *great*, *fantastic*, *happy*, *cool* and negative adjectives: *bad*, *ugly*. Then we use WordNet to predict the orientations of all the adjectives in the opinion word list. Once an adjectives orientation is predicted, it is added to the seed list. Therefore, the list grows in the process.
To implement this, we make two functions -

Orientation Search: It searches the WordNet and the seed list for each target adjective word to predict its orientation. If there exists a synonym of that adjective in the seed list, then the adjective is added to the seed list with the same orientation as that of its synonym. Otherwise, the function continues to search the antonym set of the target word in WordNet and checks if any antonym is in the seed list i.e has known orientation. If so,

the target orientation is set to the opposite of the antonym and the adjective is added to the seed list.

Orientation Prediction: This function repeated calls the *Orientation Search* function with the opinion list and seed list as parameters. It checks the size of the seed list before and after calling the *Orientation Search*. It repeats the process until both the sizes come out to be same i.e no word was included in the seed list in that call of the *Orientation Search*. The reason why *Orientation Search* is called many times is that it may be possible that a word's orientation may be found in a later call of the procedure with an updated seed list.

For those adjectives that WordNet cannot recognize, they are discarded as they may not be valid words and for those that we cannot find orientations, we assign them a neutral orientation. Also, for the case that the synonyms/antonyms of an adjective have different known semantic orientations, we use the first found orientation as the orientation for the given adjective.

We save the positive, negative and neutral oriented opinions separately to use in the next section.

3.7 Opinion Matrix Formation

Since, currently our opinion features list is quite large, we must use some kind of reduction in the number of features to make the opinion matrix for evaluation.

First we decide to take the most frequently appearing nouns/noun phrases(features). This is implemented by taking the nouns from the saved positive, negative and neutral opinions list and sorting them in reverse order of their frequency in the entire text of reviews. We then take the 20 most common nouns from the sorted list.

Next we group the various opinions in each of positive, negative and neutral opinions list by their noun/noun phrase term. Now for each noun in the most frequent noun list we get the corresponding list of opinions from each of the positive, negative and neutral opinion lists. The lists are taken as the different columns of the opinion matrix. The members of each such list of opinion features collectively form one column of the opinion matrix.

Now for each review we check if it contains any noun from the most frequent noun list, if so, we check if it also contains any of the opinions associated with that noun. If both the conditions are satisfied then we mark a '1' in the column corresponding to that list of opinions, otherwise put a '0'.

Doing this finally results in the opinion matrix for the product.

Now we make 3 different opinion matrices for each product. The first matrix is the original matrix with columns as defined above. we name it s_orig_A_5. We make another matrix which does not contain those opinions as columns which have frequency equal to 1 in the entire datset *i.e.* are quite rare though may be important. We name this matrix as s_wo_1_A_5. Our third opinion matrix s_wo_2_A_5 does the same for the opinions with frequency 2 or lower.

Opinion Matrix	**Metrics**						
	mth	**cos_d**	**cos**	**cpr**	**a-dcg**	**unwt**	**recall**
s_orig_A_5	0.64	0.87	0.87	0.56	4.61	15.81	0.76
s_wo_1_A_5	0.65	0.86	0.86	0.54	4.60	15.85	0.75
s_wo_2_A_5	0.63	0.86	0.87	0.56	4.60	15.97	0.75

Table 1: Results for different metrics

4 Results

The evaluation was done on the three opinion matrices using different metrics, the results of which are shown in Table-1. The little to no variation in the results suggest that removing the opinions with frequency one or two didn't have much effect on the results. Also the fact that the original matrix has the highest value among the three matrices in all the metrics suggests that removing those opinions with frequency one or two had negative effect on the reults and hence those opinions were important and shouldn't be neglected.

References

Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules.

Junsheng Cheng Bing Liu, Minqing Hu. 2005. Opinion observer: analyzing and comparing opinions on the web. *WWW '05*.

Philip S. Yu Bing Liu, Xiaowen Ding. 2008. A holistic lexicon-based approach to opinion mining. *WSDM '08*.

Yiming Ma Bing Liu, Wynne Hsu. 1998. Integrating classification and association rule mining. *KDD'98*.

Frans Coenen. The lucs-kdd implementations of the cba algorithm. http://cgi.csc.liv.ac.uk/~frans/KDD/Software/CBA/cba.html.

Minqing Hu and Bing Liu. 2004a. Mining and summarizing customer reviews. *KDD'04*.

Minqing Hu and Bing Liu. 2004b. Mining opinion features in customer reviews. *AAAI'04*.

Miller K. Fellbaum C. Gross D. Miller, G. and R. Beckwith. 1990. Introduction to wordnet: An on-line lexical database.

Janyce M. Wiebe Vasileios Hatzivassiloglou. 2000. Effects of adjective orientation and gradability on sentence subjectivity.