Volume 5, Issue 6, June 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Overview And Preliminary Results For Opinion Mining Using Fuzzy String Searching

Prerna Tekchandani
Research Scholar, Department of CSE,
RITEE, Raipur, India

Avinash Dhole
Associate Professor, Department of CSE,
RITEE, Raipur, India

Abstract— One of the important types of information on the Web is the opinions expressed in the user generated content, e.g., customer reviews of products, forum posts, and blogs. In this paper, we focus on customer reviews of products. In particular, we study the problem of determining the semantic orientations (positive, negative or neutral) of opinions expressed on product features in reviews. This problem has many applications, e.g., opinion mining, summarization and search. Here we utilize a list of opinion words(also called opinion lexicon) for the purpose. Opinion words are words that express desirable (e.g., great, amazing, etc.) or undesirable (e.g., bad, poor, etc.) states. Sometime customers writes the wrong spellings for the product property so we solve this problem here by using fuzzy string searching. Approximate string matching (often colloquially referred to as fuzzy string searching) is the technique of finding strings that match a pattern approximately (rather than exactly).

Keywords— opinion mining, sentiment analysis, Fuzzy String Searching, Levenshtein distance, Semantic orientation.

I. INTRODUCTION

Opinion mining can be defined as a sub-discipline of computational linguistics that focuses on extracting people's opinion from the web. The recent expansion of the web encourages users to contribute and express themselves via blogs, videos, social networking sites, etc. All these platforms provide a huge amount of valuable information that we are interested to analyse. With the rapid expansion of e-commerce over the past 10 years more and more products are sold on the Web, and more and more people are buying products online. In order to enhance customer shopping experience, it has become a common practice for online merchants to enable their customers to write reviews on products that they have purchased. With more and more users becoming comfortable with the Web, an increasing number of people are writing reviews. As a result, the number of reviews that a product receives grows rapidly. Some popular products can get hundreds of reviews or more at some large merchant sites. Many reviews are also long, which makes it hard for a potential customer to read them to make an informed decision on whether to purchase the product. If he/she only reads a few reviews, he/she only gets a biased view. The large number of reviews also makes it hard for product manufacturers or businesses to keep track of customer opinions and sentiments on their products and services. It is thus highly desirable to produce a summary of reviews. Opinion mining tasks can be divided into three main steps:

1. Using both data mining and natural language processing techniques to extract product features suggested by customers.
2. Identifying if an opinion sentence which is either positive or negative by performing three subtasks:
   - Identifying the adjectives which are used to express opinions by using natural language processing techniques
   - Determining semantic orientation (positive or negative) for each adjective
   - Deciding the opinion orientation for each sentence.
3. Summarizing the result from previous tasks.

II. RELATED WORK

Hatzivassiloglou and McKeown extracted Adjectives that occurred more than twenty times in the reviews of Wall street Journal [1]. These were then separated into good and bad lists. Adjectives that did not fit in any of these lists were dropped. Corpus was parsed to extract conjunctions between adjectives. Two clusters of semantically similar adjectives were uncovered. The cluster with higher average frequency of words was observed as positive cluster. Accuracy observed was 92%. Authors have also worked on ways to automatically identify antonyms without referring corpus for semantic description.

Bruce and Wiebe made an effort to manually tag sentences as subjective or objective by different judges and the resultant confusion matrix was analysed [2]. 14 articles were randomly chosen and every non-compound sentence was tagged. Also a tag was attached to conjunct of every compound sentence. Authors then attempted to identify if pattern exists in agreement or disagreement between human judges. Authors observed that manual tagging suffered due drawback of biased nature of human beings during tagging phase.

K. Dave *et al.* used a self-tagged corpus of sentiments [3] available on major websites such as Amazon ancnet as training set. Naïve Bayes classifier was trained as well as refined using the above mentioned corpus. The classifier was then

tested on other portion of self-tagged corpus. The sentences were parsed to check semantic correctness and then tokenised. Techniques such as co allocation substrings and stemming were applied for generalisation of tokens. N-grams (bi-gram and tri-gram) improved the results as compared to unigram. They also applied smoothing so that non-zero frequencies are available. Score were then assigned to features.

Hu and Liu crawled reviews and tagged parts of speech to extract frequent features [4]. Adjectives associated with these features were extracted and opinions for individual features in a review were determined. As the seed adjectives were already tagged as positive or negative, its associated adjectives were marked accordingly synonyms in WordNet. Using antonyms in WordNet itself other set of adjective was constructed. For example for a positive seed adjective like good, excellent, outstanding the set formed by synonyms were all marked as positive adjectives and the antonyms of these like bad and disgusting formed the other set of adjectives that expressed negative opinions.

Beineke *et al.* deployed model that fits Naive Bayes classifier in Turneys model [5]. Five positive and negative anchor words were chosen and each list was expanded using Word Net. Classifier was trained using a small tagged corpus and model was refined and applied on test data.

Andreevskaia and Bergler proposed method for extracting adjective that have opinions from WordNet [6]. Corpus based as well dictionary based approaches were used for extracting opinionated adjectives. Dictionary based approach was used to partially disambiguate the results of parts of speech tagger.

Esuli and Sebastiani developed a WordNet of sentimental words called as SentiWordNet [7]. The lexicon was build using eight ternary classifiers. Every WordNet synset was classified as positive, negative or objective, resulting into fine-grained and exhaustive Lexicon. A small subset of training data was manually labeled. The remaining training data was iteratively labeled or trained using small subset of labeled training set. Two classification algorithms (Rocchio learner and SVM) were applied on four subsets of training data to build above mentioned ternary classifier. Three resultant classes generated are positive, negative or objective. Positivity, negativity or objectivity was represented using a graded scale.

Aciar *et al.* converted text data i.e. unstructured data to Ontology i.e. structured data [8]. Along with product quality information even review quality information was maintained such as reviewer's experience. Tagged set of rules are maintained. Individual sentence was parsed and the feature in the sentence was extracted and set of rules were used to mark the feature as positive or negative. More importance was given to experienced reviewers text.

Takama and Muto generated user profiles from user's television watching behaviour [9]. Various ways to identifying user opinion towards a TV show are discussed such as TV watching time. These were estimated by the utterances made by a person while watching TV. Not only the popularity of the show was calculated but the persons profile was build. Comments were recorded from the sample set instead of utterance made by individual to avoid limitation of speech to text software.

Pang and Lee have described the problem of Sentiment analysis as classification at multiple levels [10]. Text was first classified as opinionated or informative. If the given text was already present in more traditional fact-based analysis. A discussion of available resources, benchmark datasets, and evaluation campaigns is provided.

Subrahmanian and Reforgiato graded sentiments by the combination of adjective, verb and adverb [11]. In contrast to the algorithms that extracted the sentiments using adjective - verb combination or adverb - adjective combination, the model was trained using adjective, verb and adverb combination. The opinion was drawn from eight combinations of positivity or negativity of adjective, verb and adverbs in the reviews.

Zhang constructed computational model that explored reviews linguistics properties to judge its usefulness [12]. Support Vector Regression (SVR) algorithm was used for classification. In contrast to major studies which filter out subjective information in any review or are not considered important, Zhang claimed that the quality of review was reasonably good if it was a good combination of subjective and objective information.

Sindhwani and Melville proposed a general framework for incorporating lexical information as well as unlabeled data within standard regularized least squares for sentiment prediction tasks [13]. Joint sentiment analysis of documents and words was targeted based on a bipartite graph representation of the data.

Denecke proposed an approach to multilingual sentiment classification that combined standard translation software and sentiment analysis resources for English [14].

Cai *et al.* stated that solution for sentiment analysis should include a sentiment classification scheme as well as a sentiment topic detection scheme [15]. The sentiment classification component measured the relative sentiment (on a positive/negative scale) expressed by the words. The sentiment topic detection component detected the most significant topics hidden behind each sentiment category using a combined Point-wise Mutual Information and word support metrics.

Chenutilised a rich feature set to represent forum communications, and machine learning techniques to identify and measure the sentiment polarity [16].

Bermingham *et al.* maintained user profile and interactions in a targeted group by mining their opinions on the social networking sites [17].

Nicholls and Song used Parts-of-Speech based method for weighting terms [18]. Adjectives and adverbs were found to be nearly up to times relevant as compared to nouns and verbs.

Dang *et al.* trained a model by a training set of positive and negative tagged reviews [19]. Pre-processing portion included data collection using a spider program and cleaning (html tags) using parser. This cleaned data was stored in relational databases. Major features considered for sentiment classification were context free, context specific and sentiment features. Dictionary verbs. Authors observed that adding context free and context specific features improved

sentiment classification performance.

Wu *et al.* addressed visualization related issues in opinion mining [20]. The results was aggregated and summarized to visualize in different ways. Authors constructed an interactive visualization system. A set of good and bad word list for hotel

III. PROPOSED TECHNIQUE

The framework of the proposed approach is illustrated in Figure 1. In this approach, a review will be an input text. A review may consist of a word, a sentence, or a paragraph. It is common to find a reviewer to write just one word, such as "great", "excellent" or "useless". A reviewer may also write his/her opinion in one sentence, for example, "The room is great". Some reviewers may take more effort when writing are view. Their opinions are expressed in more than one sentence. Most of the time it will be a paragraph. In mining opinion, although a reviewer just says one word, it will be taken as a review.

There are 7 steps in our proposed technique. First we take our review file and remove any special character to make data clean and clear for processing. Then we split the data and take only words to compare in wordlist. We have to create a bucket of product properties, positive and negative vocabulary. Then we compare the found words from file to the product properties. We process the one word property and more than one word property differently. Applying fuzzy string searching on the product properties. Sometimes the customer writes the review and misspell the product properties therefore here we use approximate string searching so that if customer misspell any word then also our software recognize that word and do the further processing. Now we check whether the found word is positive or negative by comparing with positive and negative vocabulary. If the word found is positive then we add +1 to the review count . And if the word found is negative then we add -1 to the review count. Then we will add all the review count for the particular product. After that we visualize this result by a bar graph. Bar graph shows the total review count for all the processed product. By the bar graph we can compare the various products on the basis of their review count.

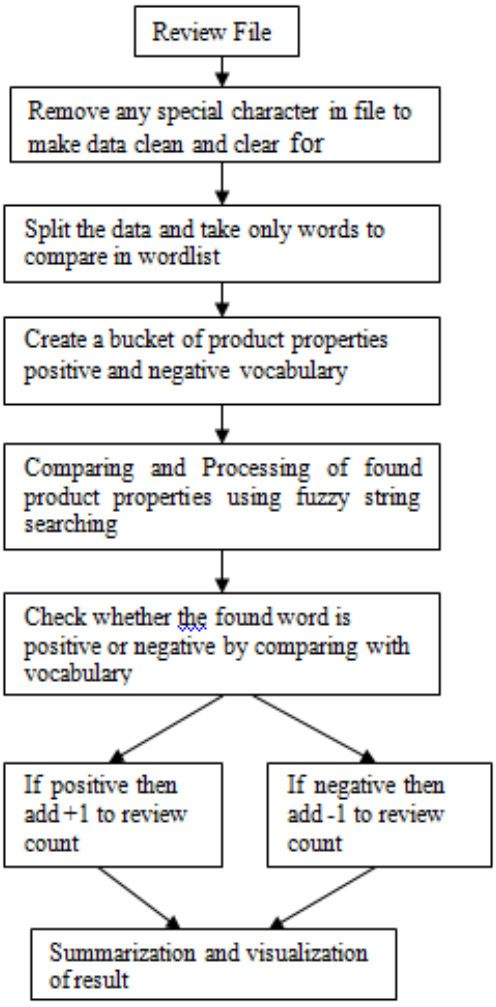

Fig. 1. The Approach Framework

IV. FUZZY STRING SEARCHING

In computer science, approximate string matching (often colloquially referred to as fuzzy string searching) is the technique of finding strings that match a pattern approximately (rather than exactly). The problem of approximate string matching is typically divided into two sub-problems: finding approximate substring matches inside a given string and finding dictionary strings that match the pattern approximately.

A. Overview

The closeness of a match is measured in terms of the number of primitive operations necessary to convert the string into an exact match. This number is called the edit distance between the string and the pattern. The usual primitive operations are

- insertion: cot → coat
- deletion: coat → cot
- substitution: coat → cost

These three operations may be generalized as forms of substitution by adding a NULL character (here symbolized by *) wherever a character has been deleted or inserted:

- insertion: co*t → coat
- deletion: coat → co*t
- substitution: coat → cost

Some approximate matchers also treat transposition, in which the positions of two letters in the string are swapped, to be a primitive operation. Changing cost to cots is an example of a transposition.

Different approximate matchers impose different constraints. Some matchers use a single global unweighted cost, that is, the total number of primitive operations necessary to convert the match to the pattern. For example, if the pattern is coil, foil differs by one substitution, coils by one insertion, oil by one deletion, and foal by two substitutions. If all operations count as a single unit of cost and the limit is set to one, foil, coils, and oil will count as matches while foal will not.

Other matchers specify the number of operations of each type separately, while still others set a total cost but allow different weights to be assigned to different operations. Some matchers permit separate assignments of limits and weights to individual groups in the pattern.

B. Levenshtein distance

In information theory and computer science, the Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. It is named after Vladimir Levenshtein, who considered this distance in 1965.

Levenshtein distance may also be referred to as edit distance, although that may also denote a larger family of distance metrics. It is closely related to pairwise string alignments.

Mathematically, the Levenshtein distance between two strings a, b is given by $\operatorname{lev}_{a,b}(|a|,|b|)$ where

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

where $a_i = b_j 1_{(a_i \neq b_j)}$ is the indicator function equal to 0 when and equal to 1 otherwise.

Note that the first element in the minimum corresponds to deletion (from a to b), the second to insertion and the third to match or mismatch, depending on whether the respective symbols are the same.

C. Example

For example, the Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

kitten → sitten (substitution of "s" for "k")
sitten → sittin (substitution of "i" for "e")
sittin → sitting (insertion of "g" at the end).

V. RESULT ANALYSIS

This section of paper provides result after implementation of opinion mining algorithm is discussed and presented. In our experiment we took the reviews from different E-commerce website like Amazon, Flipkart etc. Here we took reviews of different mobile phones. The mobile phones whose reviews collected are “Apple iphone 4S”, “Apple iphone 5S”,”Moto G2”,”Moto X2”,”Moto E2”,” HTC desire”, “Lumia 535”.We collected the properties of mobile phones. We also collected all the positive and negative words. We processed these reviews to find the positive and negative opinions of the users using fuzzy search algorithm. Then summarize the result by calculating the efficiency of our algorithm.

In our opinion mining algorithm we used “levenshtein distance” to calculate fuzzy Score for matching two strings.To get appropriate result, we have initialised Fuzzyness by 0.8 . A value of 0.8 means that the difference between the word to find and the found words is less than 20%

Fuzzy Score can be calculated using following formula.

Fuzzy Score = 1.0 - levenshteinDistance / length of string

…equation 1

Where Levenshtein distance between two strings is given by

Fuzzy Score >= Fuzzyness …equation 2

The found Fuzzy score of string matched with fuzzyness and it should be greater than or equal to 0.8 The fuzzyness of algorithm can be varry from 0.0 to 1.0 which should be predefined by the programmer.
The calculated fuzzy score of different properties are shown below in the table. The Levenshtein Distance is calculated by equation 1 and fuzzy score is calculated by equation 2. Length of the string is the length of the original string.
Below table shows the levenshtein diastance found and the fuzzy score :

Table 1. Calculated fuzzy score of different properties of mobile phones

S.No	Properties	Found Properties	Levenshtein Distance	Length of string	Fuzzy Score
1	Battry	Battery	1	7	0.857
2	Camra	Camera	1	6	0.833
3	Disply	Display	1	7	0.857
4	Prise	Price	1	5	0.800
5	Procesor	Processor	1	9	0.888
6	Speker	Speaker	1	7	0.857
7	Scren	Screen	1	6	0.833
8	Weght	Weight	1	6	0.833
9	Desin	Design	1	6	0.833
10	Softwar	Software	1	8	0.875

Below tables shows the summarized result and compare our method fuzzy search algorithm to normal searching technique.

Table 2. Found Properties by fuzzy search and normal search

Reviews	Product Name	Found Properties By Fuzzy Search	Found Properties By normal search
Review 1	Apple Iphone 4S	Battery, Ratina display, Sound quality, Touch interface, Camera	Ratina display, Sound Quality, Touch interface
Review 2	Apple Iphone 5S	Display, RAM, Camera, Call Quality, Screen	Call Quality camera, RAM
Review 3	Moto G2	Price, Processor ,Call Quality, Camera, Screen, Speaker, Audio codec, Battery	Call quality, audio codec, Price, Screen
Review 4	Moto X2	Price, Weight, Camera, Sound Quality, processor	Weight,camera, Sound quality
Review 5	Moto E2	Grip, Speaker, Sound Quality, Camera	Grip,Build quality
Review 6	HTC Desire	Processor,Design,Camera,Software Display,Screen,Sound,battery,RAM	Processor,Camera,screen, Sound, battery, RAM
Review 7	Microsoft Lumia	Front camera, Design, speaker, display, call quality,battery, price	Front camera, call quality, battery price

Table 3.Overall analysis of result

Reviews	Product Name	Number of found properties by Fuzzy search	Number of found properties by normal search
Review 1	**AppleIphone 4S**	**5**	**3**
Review 2	**AppleIphone 5S**	**5**	**3**
Review 3	**Moto G2**	**8**	**4**
Review 4	**Moto X2**	**5**	**3**
Review 5	**Moto E2**	**4**	**2**
Review 6	**HTC Desire**	**9**	**6**
Review 7	**Microsoft Lumia**	**7**	**4**

Overall in 7 reviews of different mobile phones we found the following number of properties in fuzzy search and normal search respectively
Total number of properties found by Fuzzy search = 43
Total number of properties found by normal search = 25

The effieciency of our algorithm is calculated by the following formula:

$$\text{Efficiency} = \frac{\text{No of words found by fuzzy search} - \text{No of words found by normal search}}{\text{No of words found by normal search}}$$

$$\text{Efficiency} = \frac{43-25}{25} = 72\%$$

This is the efficiency obtained in our proposed technique. Improved efficiency is obtained by employing fuzzy search technique to opinion mining. The efficiency of the proposed system would increase with the increase in the number of found properties by fuzzy search. And the number of found words will increase if we decrease the fuzzyness percentage which could cause different unexpected results sometimes.

V. CONCLUSION

This paper proposed an effective method for identifying semantic orientations of opinions expressed by reviewers on product features. It is able to deal with a major problem occurred while opinion mining i.e when customer misspell any product properties. We handle this problem here by using fuzzy string searching. The future work of this research is to enlarge the positive and negative vocabulary, evaluate the proposed approach using more reviews and use context knowledge to determine a sentiment phrase in a review.

REFERENCES

[1] V. Hatzivassiloglou and K. R. McKeown, "Predicting the Semantic Orientation of Adjectives," In Proc of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conf. of the European Chapter of the Association for Computational Linguistics, pp 174-181, 1997.

[2] R. F. Bruce and J. M. Wiebe, "Recognizing Subjectivity: A Case Study in Manual Tagging," *In Natural Language Engineering, ACM, vol. 05,issue 02*, pp 187-205, 1999.

[3] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews," *InProc of the 12th Int'l Conf. on World Wide Web*, pp 519 - 528, 2003.

[4] M. Hu and B. Liu, "Mining and Summarizing Customer Reviews," *In Proc of the 10th Int'l Conf. on Knowledge discovery and data mining*, pp 168-177, 2004.

[5] P. Beineke, T. Hastie, and S. Vaithyanathan, "The Sentimental Factor: Improving Review Classification via Human-provided Information," *In Proc of the 42nd Annual Meeting on Association for Computational Linguistics*, Article No. 263, 2004.

[6] Andreevskaia and S. Bergler, "Mining WordNet for Fuzzy Sentiment: Sentiment Tag Extraction from WordNet Glosses," *In Proc of the 11thConf* 209-216, 2006.

[7] Esuli and F. Sebastiani, "Senti-WordNet: A Publicly Available Lexical Resource for Opinion Mining," *In Proc of the 05th Conf. on LanguageResources and Evaluation,* pp 417 – 422, 2006.

[8] S. Aciar, D. Zhang, S. Simoff, and J. Debenham, "Informed Recommender: Basing Recommendations on Consumer Product Reviews," *In Intelligent Systems, IEEE, vol.22, issue no.03*, pp.39-47, 2007.

[9] Y. Takama and Y. Muto, "Profile Generation from TV Watching Behavior Using Sentiment Analysis," *In Proc. of Int'l Conf. on WebIntelligence and Intelligent Agent Technology - Workshops*, pp.191-194,2007.

[10] Pang and L. Lee, *"Opinion Mining and Sentiment Analysis,"* Foundations and Trends in Information Retrieval, vol. 02, issue nos. 1/2, pp. 1-114, 2008.

[11] V. S. Subrahmanian and D. Reforgiato, "AVA: Adjective-Verb-Adverb Combinations for Sentiment Analysis," *In Intelligent Systems, IEEE,vol.23, issue no 04*, pp.43-50, 2008.

[12] Z. Zhang, "Weighing Stars: Aggregating Online Product Reviews for Intelligent E-commerce Applications," *In Intelligent Systems, IEEE, vol.23, issue no.05*, pp.42-49, 2008.

[13] V. Sindhwani and P. Melville, "Document-Word Co-regularization for Semi-supervised Sentiment Analysis," *In Proc. of Int'l Conf. on DataMining,* pp 1025-1030, 2008.

[14] K. Denecke, "Using SentiWordNet for multilingual sentiment analysis," *In Proc. of 24th Int'l Conf. on Data Engineering Workshop*, pp 507 –512, 2008.

[15] K. Cai, S. Spangler, Y. Chen, and L. Zhang, "Leveraging Sentiment Analysis for Topic Detection," *In Proc. of Int'l Conf. on WebIntelligence and Intelligent Agent Technology,* vol. 01, pp.265-271,2008.

[16] H. Chen," Sentiment and affect analysis of Dark Web forums: Measuring radicalization on the internet," *In Proc. of Int'l Conf. onIntelligence and*

[17] A. Bermingham, M. Conway, L. McInerney, N. O'Hare, and A. Smeaton, "Combining Social Network Analysis and Sentiment Analysis to Explore the Potential for Online Radicalisation," *In Proc. of Int'lConf. on Advances in Social Network Analysis and Mining*, pp 231-236,2009.

[18] C. Nicholls and F. Song, "Improving Sentiment Analysis with Part-of-Speech Weighting," *In Proc. of Int'l Conf. on Machine Learning andCybernetics*, pp 1592-1597, 2009

[19] Y. Dang, Y. Zhang, and H. Chen, "A Lexicon-Enhanced Method for Sentiment Classification: An Experiment on Online Product Reviews," *In Intelligent Systems, IEEE, vol. 25,issue no.4*, pp.46-53, 2010.

