Aspect-level Sentiment Analysis

Yuanxu Wu
New York University
yw2983@nyu.edu

Guanquan Dong
New York University
gd1084@nyu.edu

Xiaodan Wang
New York University
xw1645@nyu.edu

I. Introduction and Related Work

Comparing with the sentence-level and document-level sentiment analyses, aspect-level (also called fine grained sentiment analysis) can provides the specific opinions or polarity about the different aspects of entities. It is often desired in practical shopping applications and websites.

Instead of simply addressing the sentiment of a review, aspect-based sentiment analysis will extract possible features in the review and justify their own polarities. For example, in the sentence "The software of Canon G3 is awful but the picture is great." The aspect-based sentiment analyses will get two features "software" and "picture", and software's polarity is negative whereas the picture's polarity is positive. The figure 1-1 is a screenshot from Google shopping, it shows the result of Samsung Galaxy S8, the overall sentiment review is 4.7 and the fine grained aspects such as "Larger Screen" with a positive polarity, "Longer Battery Life" with a positive polarity and "Heavier Phone" with a negative polarity.

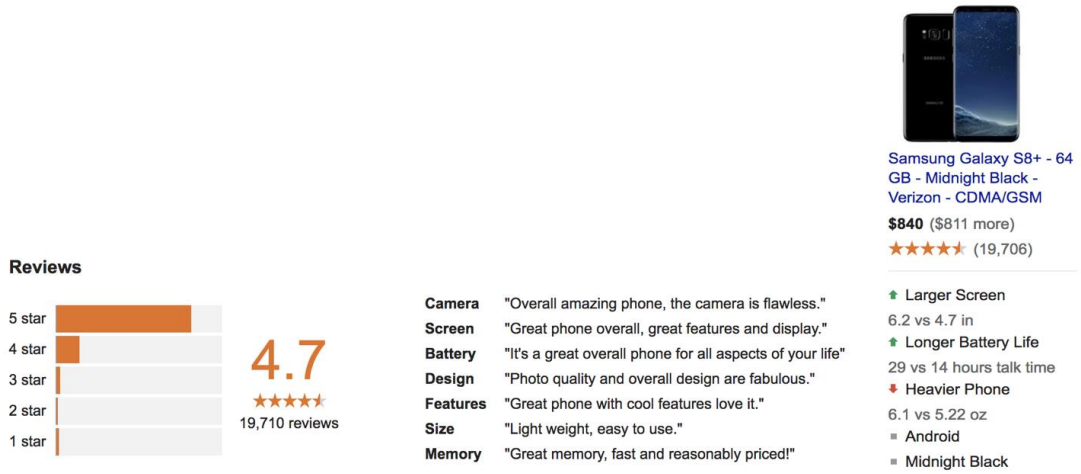


Figure1-1

We use the Canon G3 review data from Hu and Liu (2004)[1], and manually select a small opinion words set from the review data with polarity (positive, negative, or neutral) assigned for each opinion word as the opinion seed for our program.

We employ the Stanford POS tagging tool[2] and Stanford CoreNLP dependency parser[3] as our POS tagging and dependency parsing tool.

1 https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
2 https://nlp.stanford.edu/software/tagger.shtml
3 https://stanfordnlp.github.io/CoreNLP/

Hu and Liu (2004) first proposed a method to extract product aspects based on association rules. The idea can be summarized briefly by two points: (1) Identify frequent nouns and noun phrases as frequent aspects. (2) Using relations between aspects and opinion words to identify infrequent aspects.

Zhuang et al. (2006) employed the dependency relation to extract <aspect,opinion> pairs from movie reviews. The idea of using the modifying relationship of opinion words and aspects to extract aspects can be generalized to using dependency relation, and that is the fine grained sentiment analysis based on dependency parsing. It is a rule-based system, and one possible shortcoming of this approach is that the recall rate will highly depends on the keyword list which is captured main feature/opinion words in movie reviews. Thus, it requires a large labeled training set to get the key word list for feature and opinion word.

Our project is implement based on idea from the published paper from Wang and Wang (2008) and Qiu et al. (2011). The Double propagation method needs only an initial set of opinion word seed as the input. The propagation rule is generated based on the observation that opinions usually have targets. Furthermore, it also found that opinion words have relations among themselves and so do targets among themselves too, as discussed Zhang and Liu's book(2014). A key advantage of this approach is that it only needs a small opinion seed to start the bootstrapping process, and it is a semi-supervised machine learning approach.

Our project modified the Double Propagation algorithm given in Qiu et al. (2011) 's paper, by changing the input from one review about a product to all reviews about a product, thus the double propagation process will iterate between all reviews within the domain of a product. This will help to reduce the rely on the quality of the input opinion seed in order to get a high recall rate for the features. Due to this modification, we also change the polarity rules discussed in Qiu's paper, as we need it to be fit for our system, particularly by determining in which case, the polarity should be transfer and copy, reset to neutral or updated by average of new polarity and old polarity among different reviews for a single product. Details about these modifications are discussed in the later section. Except these modification, we simply adopt the 4 propagation rules and the general idea behind the double propagation proposed by Qiu et al.(2011).

II. The Double Propagation Algorithm

In this algorithm, a small opinion words seed set with polarity for each word, *O* and reviews from all users about a product, *R* are used as the input. The reviews R normally contains multiple reviews from many users. The objective of this algorithm is to extract new features and opinion iteratively based on the dependency parsing relationship and POS of word in the form of a double propagation process. The extraction method is rule-based, and the rules will be discussed later in details in next session. In general, the idea of the algorithm is first to extract new opinion word and features based on the seed opinion word set, and then use the newly extracted opinion words and features to extract other opinion words and features. The algorithm stops when there is no new opinion word and feature can be extracted from the review. In this way, even if the starting opinion words seed set is small, this method is still able to get a high recall and at the same time, the opinion sentiment associated with the feature are also extracted.

A walk-through example is demonstrated as below.
Input:

opinion word seed: {<*great : +2*>}
a user's review about Canon G3: {"*Canon G3 takes great picture. The picture is amazing. You may need to get more storage to store high quality picture and record movie. The software is also amazing*"}.

Output:

{<picture : +2>, <amazing: +2>, <software: +2>,<movie: +2> }

In the first iteration, with the only seed <great:+2>, based on rule 1, we can extract new feature “picture”, and its polarity is assigned to that of “great” based on the polarity rule. Then based on rule 2, we can extract “amazing” given “picture”, and based on rule 3, we can extract “movie” given “picture”. In the second iteration, based on rule 1, “software” is extracted given “amazing”. As we can see, from a single opinion seed “great”, we can extract 3 new features and 1 new opinion word, the polarities are also updated based on polarity rules.

The persudo code of the double propagation algorithm:

```
Function DoubleProp(opinion_seed_set, all_reviews_for_a_product) {
        opinion_expand_set = opinion_seed_set
        all_feature_set = empty

        while (True) {
                foreach one_review in all_reviews_for_a_product
                        new_feature_i_set = empty
                        new_opinion_i_set = empty
                        foreach sentence in one_review

                                //Rule 1.1
                                temp_new_feature_R11 = Rule11(sentence, opinion_expand_set)
                                foreach feature in temp_new_feature_R11
                                        if feature is new
                                                feature.polarity = opinion_word.polarity
                                                add feature to new_feature_i_set
                                        else
                                                update feature.polarity based on polarity rule

                                //Rule 1.2
                                temp_new_feature_R12 = Rule12(sentence, opinion_expand_set)
                                foreach feature in temp_new_feature_R12
                                        if feature is new
                                                feature.polarity = opinion_word.polarity
                                                add feature to new_feature_i_set
                                        else
                                                update feature.polarity based on polarity rule

                                //Rule 4.1 and Rule 4.2.
                                //For rule 4, the new opinion checking and polarity updating is implemented inside Rule41 and
                                Rule42 function.
                                temp_new_opinion_R41 = Rule41(sentence, opinion_expand_set)
                                temp_new_opinion_R42 = Rule42(sentence, opinion_expand_set)
                                add each new_opinion in temp_new_opinion_R41 and temp_new_opinion_R42 into
                                new_opinion_i_set


                        all_feature_set = all_feature_set + new_feature_i_set
                        opinion_expand_set = new_opinion_i_set
                        new_feature_j_set = empty
                        new_opinion_j_set = empty


                        foreach sentence in one_review

                                //Rule 2.1
                                temp_new_opinion_R21 = Rule21(sentence, new_feature_i_set)
                                foreach opinion in temp_new_ opinion_R21
                                        if opinion is new
                                                opinion.polarity = opinion_word.polarity
                                                add opinion to new_opinion_j_set
                                        else
                                                update opinion.polarity based on polarity rule

                                //Rule 2.2
                                temp_new_opinion_R22 = Rule22(sentence, new_feature_i_set)
                                foreach opinion in temp_new_ opinion_R22
                                        if opinion is new
```

```
                                        opinion.polarity = opinion_word.polarity
                                        add opinion to new_opinion_j_set
                                    else
                                        update opinion.polarity based on polarity rule

                                //Rule 3.1 and Rule 3.2.
                                //For rule 3, the new feature checking and polarity updating is implemented inside Rule31 and
                                Rule32 function.
                                temp_new_feature_R31 = Rule31(sentence, new_feature_i_set)
                                temp_new_feature_R32 = Rule32(sentence, new_feature_i_set)
                                add each temp_new_feature_R31 and temp_new_feature_R32 into new_feature_j_set

                            new_feature_i_set = new_feature_i_set + new_feature_j_set
                            new_opinion_i_set = new_opinion_i_set + new_opinion_j_set
                            all_feature_set = all_feature_set + new_opinion_j_set
                            opinion_expand_set = opinion_expand_set + new_opinion_j_set

                            if new_feature_i_set is empty and new_opinion_i_set is empty
                                break
        }
}
```

III. Propagation Rules Defined Based on Relations

In our double propagation methods, there are four subtasks: (1) extract target words based on opinion words; (2) extract opinion words based on extracted target words (3) extract target words based on extracted target words; (4) extract opinion words based on both the given and the extracted opinion words. We summarized four types of rules based on these four subtasks.

In the following sections, *o* (or *t*) stands for opinion (or target) word. {*O*} (or {*T*}) stands for the opinion (or target) words set, either given or extracted. *H* means any word. *POS(O)* (or *POS(T)*) stands for the part-of-speech information of the word. *O-dep* (or *T-dep*) stands for the dependency relation of the word *O* (or *T*). {*JJ*} stands for the POS tags set of potential opinion words, containing *JJ, JJS, JJR*. {*NN*} stands for the POS tags set of potential target set, containing *NN, NNS, NNP*. {*MR*} stands for the dependency relations describing relations between opinion words and target words, consist of {*amod, nmod, pnmod, nsubj, s, dobj, iobj, desc, xcomp*}. {*CONJ*} stands for the dependency relations between opinion (or target) words and opinion (or target) words, contains *conj* only. The arrows represent dependency relations between two words, such as $O \rightarrow O\text{-}dep \rightarrow T$ means O depends on T through a dependency relation *O-dep*. "==" represents the two dependency relations are equivalent. Here equivalent specifically means both modify relations (contains *amod, nmod, pnmod*) or both objective or subjective relations (contains *dobj, iobj, nsubj*).

Rule one: extract target words using opinion words

There are two sub-rules in rule one. The observation equations are shown below.
$R1_1$: $O \rightarrow O\text{-}dep \rightarrow T$ *s.t.* $O \in \{O\}$, $O\text{-}dep \in \{MR\}$, $POS(T) \in \{NN\}$
$R1_2$: $O \rightarrow O\text{-}dep \rightarrow H \leftarrow T\text{-}dep \leftarrow T$ *s.t.* $O \in \{O\}$, $O\text{-}dep, T\text{-}dep \in \{MR\}$, $POS(T) \in \{NN\}$

In rule 1-1, if an opinion word (*O*) depends on a word (*T*) through a dependency relation in {*MR*} set and the POS tag of this word (T) is in {*NN*} set, then we treat this word (*T*) as a new target.

In rule 1-2, if an opinion word *(O*) depends on a random word *(H*) through a dependency relation in {*MR*} set and another word (*T*) depends on this same random word (*H*) through a dependency relation in {*MR*} set and the POS tag of this word (*T*) is in {*NN*} set, then we treat this word as a new target (*T*).

In addition, this new target will be assigned the polarity of this opinion word.

For example,
The phone has a good screen.
(('has', 'VBZ'), 'nsubj', ('phone', 'NN'))
(('phone', 'NN'), 'det', ('The', 'DT'))
(('has', 'VBZ'), 'dobj', ('screen', 'NN'))
(('screen', 'NN'), 'det', ('a', 'DT'))
(('screen', 'NN'), 'amod', ('good', 'JJ'))
good → 'amod' → screen

iPod is the best mp3 player
(('player', 'NN'), 'nsubj', ('iPod', 'NNP'))
(('player', 'NN'), 'cop', ('is', 'VBZ'))
(('player', 'NN'), 'det', ('the', 'DT'))
(('player', 'NN'), 'amod', ('best', 'JJS'))
(('player', 'NN'), 'amod', ('mp3', 'JJ'))
best → 'amod' → player ← 'amod' ← mp3

Rule two: extract opinion words using target words

There are two sub-rules in rule two. The observation equations are shown below.
$R2_1$: *O → O-dep → T s.t. T ∈ {T}, O-dep ∈ {MR}, POS(O) ∈ {JJ}*
$R2_2$: *O → O-dep → H ← T-dep ← T s.t. T ∈ {T}, O-dep, T-dep ∈ {MR}, POS(O) ∈ {JJ}*

In rule 2-1, if a word (O) depends on a target word (T) through a dependency relation in {MR} set, and the POS tag of this word (O) is in {JJ}, then we treat this word (*O*) as a new opinion.

In rule 2-2, if a word *(O*) depends on a random word *(H*) through a dependency relation in {*MR*} set and a target word (*T*) depends on this same random word (*H*) through a dependency relation in {*MR*} set and this POS tag of this word (*O*) is in {*JJ*} set, then we treat this word as a new target (*T*).

In addition, we assign the polarity of this newly retrieved opinion word (*O*) the same as the target word (*T*). If the polarity of the target word (*T*) equals to zero, then we assign the polarity of this newly retrieved opinion word (*O*) the average of all the opinion words in this sentence.

For example, the same as rule one.

Rule three: extract target using target

There are two sub-rules in rule three. As shown in the following equations:
$R3_1$: $T_{i(j)}$ *→ T-dep →* $T_{j(i)}$ *T ∈ {T}, T-dep ∈ {CONJ }, POS(*$T_{i(j)}$*) ∈ {NN}*
$R3_2$: T_i *→* T_i*-dep → H ←* T_j*-dep ←* T_j *s.t. T ∈ {T},* T_i*-dep==* T_j*-dep, POS(*T_j*) ∈ {NN}*

In rule 3-1, if a target(T_i) depends on a target (T_j) through a CONJ dependency relation and the POS tag of this word (T_j) is in {*NN*} set, then we treat this word (T_i) as a new target.

In rule 3-2, if a target *(*T_i) depends on a random word *(H*) through a dependency T_i*-dep* and another word (T_j) depends on this same random word (*H*) through a dependency relation T_j*-dep* when T_i*-dep and* T_j*-dep is the same type of dependency relation* and the POS tag of this word (T_j) is in {*NN*} set, then we treat T_i as a new target.

In addition, this new target will be assigned the polarity of this opinion word.

For example,
I like the picture and the software.
(('like', 'VBP'), 'nsubj', ('I', 'PRP'))
(('like', 'VBP'), 'dobj', ('picture', 'NN'))
(('picture', 'NN'), 'det', ('the', 'DT'))
(('picture', 'NN'), 'cc', ('and', 'CC'))
(('picture', 'NN'), 'conj', ('software', 'NN'))
(('software', 'NN'), 'det', ('the', 'DT'))
picture→ 'conj' → software

This camera has great look.
(('has', 'VBZ'), 'nsubj', ('camera', 'NN'))
(('camera', 'NN'), 'det', ('The', 'DT'))
(('has', 'VBZ'), 'dobj', ('look', 'NN'))
(('look', 'NN'), 'amod', ('great', 'JJ'))
camera→ 'nsubj' → has ← 'dobj' ← look

Rule four: Extract opinion words using opinion word

There are two sub-rules in rule one. The observation equations are shown below.
R4_1: O → O-dep → O s.t. O ∈ {O}, O-dep ∈ {CONJ}, POS(O) ∈ {JJ}
R4_2: Oi → Oi-dep → H ← Oj-dep ← Oj s.t. O ∈ {O}, O-dep, T-dep ∈ {MR}, POS(T) ∈ {NN}

In rule 4-1 and rule 4-2, we extract new opinion word with POS belongs to {JJ} based on initial opinion seed and other newly found opinion word. The new opinion word's polarity is assigned as the same as the associated opinion word.

For example, for rule 4.1
The camera is amazing and easy to use.
[((u'amazing', u'JJ'), u'nsubj', (u'camera', u'NN')),
((u'camera', u'NN'), u'det', (u'The', u'DT')),
((u'amazing', u'JJ'), u'cop', (u'is', u'VBZ')),
((u'amazing', u'JJ'), u'cc', (u'and', u'CC')),
((u'amazing', u'JJ'), u'conj', (u'easy', u'JJ')),
((u'amazing', u'JJ'), u'xcomp', (u'use', u'VB')),
((u'use', u'VB'), u'mark', (u'to', u'TO'))]

By implementing the rule 4.1, we can extract amazing from "easy -> conj -> amazing".

For rule 4.2
If you want to buy a sexy, cool mp3 player, you can choose iPod.
[((u'choose', u'VB'), u'advcl', (u'want', u'VBP')),
((u'want', u'VBP'), u'mark', (u'If', u'IN')),
((u'want', u'VBP'), u'nsubj', (u'you', u'PRP')),
((u'want', u'VBP'), u'xcomp', (u'buy', u'VB')),
((u'buy', u'VB'), u'mark', (u'to', u'TO')),
((u'buy', u'VB'), u'dobj', (u'player', u'NN')),

((u'player', u'NN'), u'det', (u'a', u'DT')),
((u'player', u'NN'), u'amod', (u'cool', u'JJ')),
((u'cool', u'JJ'), u'amod', (u'sexy', u'JJ')),
((u'player', u'NN'), u'amod', (u'mp3', u'JJ')),
((u'choose', u'VB'), u'nsubj', (u'you', u'PRP')),
((u'choose', u'VB'), u'aux', (u'can', u'MD')),
((u'choose', u'VB'), u'dobj', (u'iPod', u'NN'))]

By implementing the rule 4.2, we can extract cool from “sexy -> mod -> player -> mod -> cool”.

Polarity Assignment

We make several observations about the polarity:
Observation 1: In a review written by a single writer, the opinion about the same target usually tends to be the same if the target appears several times in the review.
Observation 2:The same opinion word tends to have same polarity in a particular domain.

Based on the two observation, we can assign polarities to newly extracted opinion word and target during the extraction process. We use an integer to represent the polarity. A positive number represent the positive polarity and the negative number represent a negative polarity. The polarity assignment is done using the following rules :

Rule 1: The polarity of the seeds will always keep the same during the whole process, whereas the polarity of other opinions word will be updated contiguously.
Rule 2: The polarity of a target is only valid in the current review where it is assigned. Beyond the current review, the polarity of the target will be initialized to 0, and be assigned again in the next review. On the other hand, the opinion word can keep its polarity instead being initialized.
Rule 3: For new opinion word (or target) extracted by known target or known opinion word, it will be assigned with the same polarity as the known one. Besides, we use a words windows to deal with the negation word. If negation word present in the word windows, the new extracted word will be assigned with the opposite polarity to the known one. The windows size is 3.
Rule 4: For the known opinion word which is not seed, when it is extracted again by another opinion word or target, their polarity will be changed to the mean value of the two polarities.
Rule 5: If an opinion word can only be found in the current review, we use the overall polarity of the review as the polarity of the opinion word.

IV. Experiments

We use one of the customer review data of Hu and Liu (2004) as the testing data. We choose the reviews of Canon G3 camera. Because of the limitation of the performance of our personal computers, we carefully chose 15 reviews from the original data (contains 45 reviews). In the review data was in following format:

[t]: the title of the review: Each [t] tag starts a review. We did not use the title information in our papers.
xxxx[+|-n]: xxxx is a product feature.
    [+n]: Positive opinion, n is the opinion strength: 3 strongest,
        and 1 weakest. Note that the strength is quite subjective.
        We only considered + and -, to test our algorithm
    [-n]: Negative opinion
##: start of each sentence. Each line is a sentence.

For example:
pictures[+3], controls[+2], battery[+2], software[+2]##it gives great pictures, the controls are easy to use, the battery lasts forever on one single charge, the software is very user-friendly and it is beautiful in it chrome casing.

Again, due to the machine computation limitation, we have to separate the 15 test reviews into 3 parts and run separately in our 3 machines. Each part will run around 2-4 hours depends on the length of the reviews. This kind of running scheme may potentially lower the recall rate, comparing to run 15 reviews together. But we can compensate this effect by expand the initial opinion seed. In our testing runs, each run starts with 30 opinion seeds selected manually with relatively high frequency.
In the original data sets labeled by Hu and Liu (2004), there are lots of phrases such as picture quality. Because our algorithm can only recognize one word as target, so we manually changed the label to one word, such as we changed picture quality to quality. In addition, there are some sentences labeled in singular format, but it appears in the sentence in plural format, then we change the label into plural format to increase the accuracy of our algorithm. Our algorithm code will save the trained reviews with actual labels and predict labels into .data file through pickle. The final score is shown below. To run the score code provided by us in terminal, just use $python3 score.py <.data file name> (For instance, $python3 score.py product_reviews_list_1.data). It will show the final output and the score of trained reviews. The real label will be shown above the review sentence, and the predict label will be shown below the sentence and '+, -' shows the sentiment of the target. For example:

pictures +3
controls +2
battery +2
software +2
it gives great pictures, the controls are easy to use, the battery lasts forever on one single charge, the software is very user-friendly and it is beautiful in it chrome casing.
software 3.0
controls -0.615234375
use 0.7395833333333334
pictures 2.625

correct target and correct polarity in this sentence (true positive):
software +
pictures +

correct target but incorrect polarity:
control –

missed target:
battery +

wrong target extracted:
use +

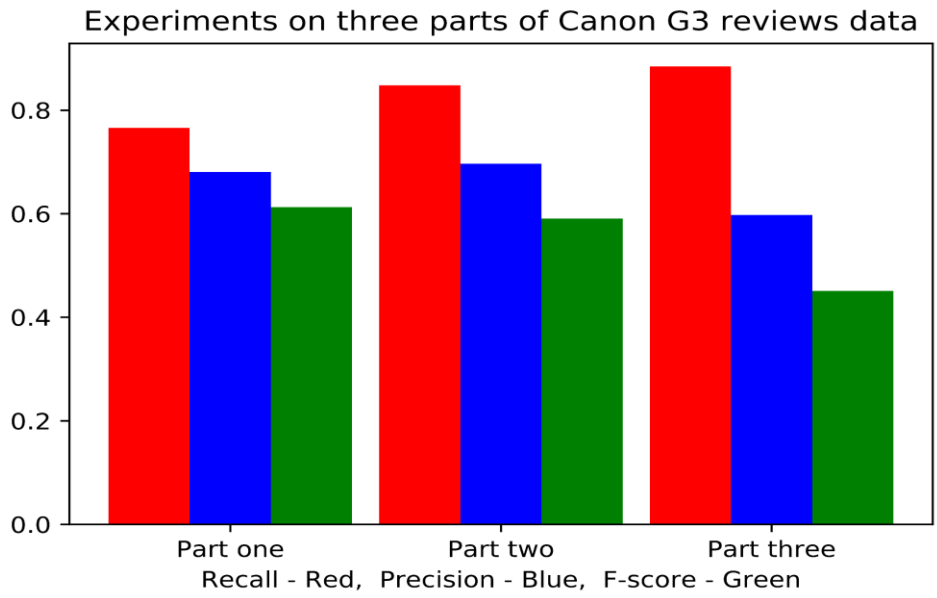
The score of all three data sets is shown below.

Part one:
true: 64, positive: 80, true positive: 49
recall=0.7656, precision=0.6125, f_score=0.6806

Part two:
true: 46, positive: 66, true positive: 39
recall=0.8478, precision=0.5909, f_score=0.6964

Part three:
true:26, positive:51, true positive: 23
recall=0.8846, precision=0.4510, f_score=0.5974

The following picture also shows the score:



V. Conclusion

In general, this language rule-based information extraction system presents a good result for aspect-level feature extraction and a great result for the feature corresponding sentiment (polarity) analysis. But we noticed that this rule-based system has its limitation, as it is highly sensitive to the contextual patterns and grammatical relations based on dependency parsing. For some reviews that do not fit the rules, the recall rate and precision is much lower than those well written and formatted reviews. So the success of the system is highly depends on domain knowledge of pattern mining and the coverage of the relation rule. At this point, we realize that real natural language processing problem is hard! Also, as stated in Zhang and Liu's book(2014), correlating with what we observed in our testing data result, the double propagation method works well for medium-sized corpuses, but for large and small corpora, it may result in low precision and low recall. The reason is that the patterns based on direct dependencies have a large chance of introducing noises for large corpora and such patterns are limited for small corpora. So Zhang et al. (2010) proposed extend approach to improve this weakness, namely aspect ranking.

Besides above lesson we learned from this project, at the very beginning stage of this term project, we also discussed the possibility of using Sequence Model, such as Hidden Markov Model approach from Jin et al. (2009a and 2009b) and Conditional Random Fields approach from Jakob and Gurevych (2010) to solve the aspect-level feature extraction and sentiment analysis problem. In the future, the sequence model and neuronal network related method will be our next exploring direction.

We also noticed that there are still much things we could do to improve this rule-based system, for example, the reference resolution case regarding "it" we encountered in the reviews, and sometimes, people just talk other similar products' good & bad in the current product's review, in order to compare these products. Our system in this case, will incorrectly captures those features and polarities of other comparing product. Again, this is another reference resolution problem. But due to the time limitation, we are not able to cover these optimizations.

References

Lei Zhang and Bing Liu, Aspect and Entity Extraction for Opinion Mining, 2014

Hu, M. and B. Liu. Mining opinion features in customer reviews. In Proceedings of National Conference on Artificial Intelligence (AAAI-2004), 2004a.

Hu, M. and B. Liu. Mining and summarizing customer reviews. In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), 2004b.

Zhuang, L., F. Jing, and X. Zhu. Movie review mining and summarization. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM-2006), 2006.

Wang, B. and H, Wang. Bootstrapping both product features and opinion words from Chinese customer reviews with cross-inducing. In Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP-2008), 2008.

Qiu, G., B. Liu, J. Bu, and C. Chen. Opinion word expansion and target extraction through double propagation. Computational Linguistics, 2011.

Jin, W. and H. Ho. A novel lexicalized HMM-based learning framework for web opinion mining. In Proceedings of International Conference on Machine Learning (ICML-2009), 2009a.

Jin, W., H. Ho, and R. K. Srihari. OpinionMiner: a novel machine learning system for web opinion mining and extraction. In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2009), 2009b.

Jakob, N. and I. Gurevych. Extracting opinion targets in a single and crossdomain setting with conditional random fields. In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2010), 2010.