An Information Retrieval-Based Approach to Determining Contextual Opinion Polarity of Words

Olga Vechtomova[1], Kaheer Suleman[2], Jack Thomas[2]

[1]Department of Management Sciences, University of Waterloo, Waterloo, ON, Canada
`ovechtom@uwaterloo.ca`
[2]Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada
`{ksuleman, j26thoma}@uwaterloo.ca`

Abstract
The paper presents a novel method for determining contextual polarity of ambiguous opinion words. The task of categorizing polarity of opinion words is cast as an information retrieval problem. The advantage of the approach is that it does not rely on hand-crafted rules and opinion lexicons. Evaluation on a set of polarity-ambiguous adjectives as well as a set of both ambiguous and unambiguous adjectives shows improvements compared to a context-independent method.

1 Introduction

Opinion detection has been an active research area in recent years. There exist a large number of approaches that attempt to identify a static sentiment polarity of words (e.g. [1-3]). It has, however, been recognized that while certain words have an unambiguous polarity, e.g. "amazing", "distasteful", others change their polarity depending on the context, e.g., "pizza was cold" vs. "beer was cold". A number of methods have been proposed to address this problem [4-7]. In [4] a supervised method was proposed to determine contextual polarity of phrases. In [5] a number of rules were used, such as conjunctions and disjunctions, manually created syntactic dependency rule templates, automatically derived morphological relationships and synonymy/antonymy relationships from WordNet. Another approach [6] used an existing opinion lexicon and a number of rules (e.g. negation rule, intra- and inter- sentence conjunction rules, synonym and antonym rules). An approach in [7] used conjunctions of ambiguous adjectives with unambiguous ones with known polarity from an opinion lexicon, and also extracted groups of related target words from Wikipedia. All of the above methods rely on rules and/or existing resources, such as WordNet or opinion lexicons. In this paper we propose an extensible framework for context-dependent polarity determination. To our knowledge this is the first method for this task, which does not rely on hand-crafted or automatically generated rules and does not utilize any pre-existing opinion vocabulary. The task of categorizing an opinion word instance into positive or negative is cast as an information retrieval problem. We build one vector of all contexts of the word *a* in the positive document set (e.g. reviews with high ratings) and another vector – of its contexts in the negative set. These vectors are treated as documents. We then build a context vector for the specific instance of *a* that we want to categorize, which is treated as the query. An IR model is then applied to calculate the query's similarity to each of the two "documents". As contexts we use dependency triples containing *a*. The approach utilizes automatically extracted lexico-syntactic contexts of the word's occurrences and their frequencies without the need to build hand-crafted rules or patterns or to use pre-existing opinion lexicons. For instance, the method in [6] has an explicit rule for conjunctives. In contrast, in our approach any conjunctives (e.g. "nice and cold"), that a word co-occurs with, say, in positive reviews, are automatically added with all other dependency triples to the positive vector of the word. In this way, the method captures a wide range of lexico-syntactic polarity clues, such as adverbial modifiers (e.g., "barely"), nouns that are targets of the opinion words, and miscellaneous syntactic constructs, such as "but" and negations. The proposed framework is extensible in a number of ways: features could be expanded (e.g., by adding other dependency triples in the sentence), filtered (e.g. by dependency relation type), or grouped by similarity. The method is evaluated on a set of adjectives with ambiguous polarity, and on another set of both ambiguous and unambiguous adjectives.

2 Methodology

Most of the product and business review sites let users assign a numerical rating representing their level of satisfaction with a product or business. In our experiments, we used a dataset of restaurant reviews, where each review has an associated rating on a scale from 1 to 10. All reviews with a rating of 10 were used as a positive training set, and all reviews with ratings 1 and 2 as negative. During the preparatory stage two vectors of context features are created for each adjective *a*. One vector *posV* is built based on the adjective's occurrences in the positive set, and the second vector *negV* is built based on its occurrences in the negative set. At the next stage, polarity of an adjective occurrence *a* in a previously unseen document *d* is determined as follows: vector *evalV* is built for this adjective based on its context within its sentence of occurrence in document *d* only. Then, a pairwise similarity of *EvalV* to the vector of the same adjective in the positive set (vector *posV*) and in the negative set (vector *negV*) is calculated.

2.1 Context feature vector construction

The following steps are performed on each of the two training sets: positive and negative. Each document in a training set is processed by using a dependency parser in the Stanford CoreNLP package. In each document, we first locate all nouns that appear as governing words in at least one dependency relation. At this stage in the algorithm, we can optionally apply a filter to process only those nouns that belong to a specific list, e.g. words denoting a specific category of review aspects (e.g. food in restaurant reviews). In our experiments we filtered the list by 456 food names which were created by using a clustering method from another project in progress. Then, for each governing word, its dependency triples with adjectives are extracted, where the dependency relation is either an adjectival modifier (amod), nominal subject (nsubj) or relative clause modifier (rcmod). An example of a dependency triple is *nsubj(pizza,* *hot)*, where “pizza” is a governor, while “hot” is a dependent word. For each adjective instance we extract all triples, in which they occur as dependent words. If one of the triples represents negation dependency relation (neg), we record that the adjective is negated. For each adjective occurrence, the following information is recorded:

- negation (1 – adjective is negated; 0 – adjective is not negated);
- dependency relation of adjective with its governing noun (amod, nsubj or rcmod);
- adjective lemma (output by Stanford CoreNLP).

These three pieces of information form adjective pattern (*AdjP*), e.g., “negation=0; amod; better”. A context feature vector is built for these patterns. The reason for building vectors for lexico-syntactic adjective patterns as opposed to just adjective lemmas, is that, firstly, we want to differentiate between the negated and non-negated instances, and, secondly, between various syntactic usages of the adjective. For instance, adjectives occurring in a post-modifier position (e.g., in “nsubj” relationship to the noun) tend to be used more in evaluative manner compared to those used in pre-modifier position (c.f: “tea was cold” and “cold tea”). While “cold tea” usually refers to a type of drink, “tea was cold” has an evaluative connotation. Also, the types of dependency relations they occur in can be different, e.g. adjectives in post-modifier position occur more with certain adverbial modifiers, which can give clues as to the adjective’s polarity, such as “barely”, “too”, “overly”, “hardly”.

Next, for each adjective instance, represented as “negation; dependency relation; lemma” adjective pattern, we extract all dependency relations that contain it. Each of them is transformed into a context feature *f* of the form: “lemma; Part Of Speech (POS); dependency relation”. For instance, if adjective “hot” occurs in dependency triple nsubj(tea, hot), the following feature is created to represent “tea” and its syntactic role with respect to the adjective: “tea, NN, nsubj”. For each feature we record its frequency of co-occurrence with the adjective pattern (used as *TF* in Eq. 1). More formally, the algorithm is described below:

Table 1. Algorithm 1: Construction of feature vectors for adjective syntactic patterns

```
 1: For each document $d \in T$
 2:    For each valid noun $n$
 3:       For each adjective $a$, dependent of $n$
 4:          If $DepRel(n,a) \in \{amod, rcmod, nsubj\}$
 5:             If any $DepRel(a,w)$ = “neg”
 6:                $negation(a)$ = 1
 7:             Else
 8:                $negation(a)$ = 0
 9:             End If
10:             Create adjective pattern $AdjP$ as “$negation(a); DepRel(n,a); lemma(a)$”
11:             For each $DepRel(a,w)$
12:                Create feature $f$ as “$lemma(w); POS(w); DepRel(a,w)$”
13:                Add $f$ to $V_{AdjP}$; Increment frequency of $f \in V_{AdjP}$
```

Where: *valid noun n* – noun that occurs in the list of nouns belonging to a specific category of review aspects (optional step); *T* – training document set, either with positive or negative review ratings (the algorithm is run separately for positive and negative document sets); *DepRel(n,a)* – dependency relation between noun *n* and adjective *a*; *DepRel(a,w)* – dependency relation between adjective *a* as either governor or dependent and any other word *w*; *POS(w)* – part of speech of *w*. V_{AdjP} – feature vector for adjective pattern *AdjP*.

Algorithm 1 is used to generate vectors for all *AdjP* patterns extracted from the positive set and, separately, from the negative set during the preparatory stage. The same algorithm is also used at the stage of determining the polarity of a specific adjective occurrence. At that stage, only the sentence containing this adjective occurrence is used to generate the vector $Eval_{AdjP}$. The pairwise similarity of $Eval_{AdjP}$ with $posV_{AdjP}$ and $Eval_{AdjP}$ with $negV_{AdjP}$ is computed. If similarity with $posV_{AdjP}$ is higher, it is categorized as positive, and as negative if similarity with $negV_{AdjP}$ is higher.

2.2 Computing similarity between vectors

We view the problem of computing similarity between vectors as a document retrieval problem. The vector ($EvalV_{AdjP}$) of a specific adjective occurrence *AdjP*, whose polarity we want to determine, is treated as the query, while the two vectors of *AdjP* ($posV_{AdjP}$ and $negV_{AdjP}$) created from the positive and negative training sets respectively, are treated as documents. For the purpose of computing similarity we use BM25 Query Adjusted Combined Weight (QACW) document retrieval function [8]. In [9] it was proposed to use it as a term-term similarity function. The $EvalV_{AdjP}$ is treated as the query, while $posV_{AdjP}$ and $negV_{AdjP}$ as documents (V_{AdjP} in Eq. 1)

$$Sim(EvalV_{AdjP}, V_{AdjP}) = \sum_{f=1}^{F} \frac{TF(k_1+1)}{K+TF} \times QTF \times IDF_f \qquad (1)$$

Where: F – the number of features that $EvalV_{AdjP}$ and V_{AdjP} have in common; TF – frequency of feature f in V_{AdjP}; QTF – frequency of feature f in $EvalV_{AdjP}$; $K = k_1 \times ((1-b)+b \times DL/AVDL)$; k_1 – feature frequency normalization factor; b – V_{AdjP} length normalization factor; DL – number of features in V_{AdjP}; $AVDL$ – average number of features in the vectors V for all *AdjP* patterns in the training set (positive or negative). The b and k_1 parameters were set to 0.9 and 1.6 respectively, as these showed best performance in computing term-term similarity in [9]. The *IDF* (Inverse Document Frequency) of the feature f is calculated as $IDF_f = \log(N/n_f)$, where, n_f – number of vectors V in the training set (positive or negative) containing feature f; N – total number of vectors V in the training set. A polarity score of *AdjP* is then calculated for both positive and negative sets as follows:

$$PolarityScore = \alpha \times Sim(EvalV_{AdjP}, V_{AdjP}) + (1 - \alpha) \times P(AdjP) \qquad (2)$$

Where: $P(AdjP)$ is calculated as number of occurrences of *AdjP* in the set (positive or negative) / total number of occurrences of all *AdjP* patterns in this set; the best result for α was 0.5. If *PolarityScore* is higher for the positive set, the polarity is positive, and if lower – negative.

3 Evaluation

For evaluation we used a corpus of 157,865 restaurant reviews from one of the major business review websites, provided to us by a partner organization. The collection contains reviews for 32,782 restaurants in the U.S. The average number of words per review is 64.7. All reviews (63,519) with the rating of 10 were used as positive training set, and all reviews with the ratings of 1 or 2 (18,713) as negative.

3.1 Evaluation on ambiguous adjectives

For this evaluation we specifically chose four adjectives (cold, warm, hot and soft) that can have a positive or negative meaning depending on the context. From reviews with ratings 3-9, we extracted all dependency triples, containing one of these adjectives in "nsubj" dependency relation with a noun representing a food name. The reason why we used "nsubj" is that post-modifier adjectives are more likely to be opinionated than pre-modifiers (i.e. related with "amod"). To select food nouns only, we applied a filter of 456 food names, created by a clustering method from another project in progress. For this experiment, we focused only on those cases that are not negated, i.e. do not occur in a dependency triple with "neg" relation. Two annotators read 888 original sentences containing these adjectives, and judged the adjective occurrences as "positive", "negative" or "objective" when they refer to food, and as "non-food modifier" for cases not referring to food. The inter-annotator agreement (Cohen's Kappa) is 0.81. There were only 2 objective cases agreed upon by the annotators, which are not included in the evaluation. The evaluation set consists of 519 positive and negative cases agreed upon by the two annotators. The cases are in the following format: "document ID; noun token; negation; dependency relation; adjective lemma; polarity". The number of positive/negative cases for "cold" is 34/180, for "warm": 29/25, for "hot": 196/10, and for "soft": 31/14.

As the baseline a context-independent method was used based on the Kullback-Leibler Divergence (KLD). KLD is used widely in IR, e.g. as a term selection measure for query expansion [10] and as a measure for weighting subjective words [11]. Polarity for each *AdjP* pattern is calculated as $P_{pos}(AdjP)*\log(P_{pos}(AdjP)/P_{neg}(AdjP))$. $P_{pos}(AdjP)$ is calculated as $F_{pos}(AdjP))/N$, where $F_{pos}(AdjP)$ is frequency of *AdjP* in the positive set, N is the total number of occurrences of all *AdjP* pattern in the positive set. $P_{neg}(AdjP)$ is calculated in the same way. Cases with $KLD>0$ are considered as positive, and with $KLD<0$ as negative. Table 2 shows Precision, Recall and F-measure for the context-based method (ContextSim) and KLD.

3.2 Evaluation on a larger set of adjectives

A larger scale evaluation was done on 606 "nsubj" and "amod" adjective patterns (482 positive and 124 negative) from 600 restaurant reviews. The dataset contains 164 distinct adjectives. The results are presented in Table 3.

While the overall improvement (F-measure) is higher for ContextSim, the precision is somewhat lower than KLD. Since the method demonstrates a much better performance on ambiguous adjectives, it makes sense to apply it only to such adjectives. We need, therefore, a method for detecting unambiguous adjectives (e.g. *excellent*) with static polarity. This is left for future work.

Table 2. Results based on a set of ambiguous adjectives.

Method	Precision	Recall	F-measure
ContextSim	0.9114	1	0.9536
KLD	0.8324	1	0.9085

Table 3. Results based on adjectives from 600 reviews.

Method	Precision	Recall	F-measure
ContextSim	0.8874	0.967	0.9255
KLD	0.9185	0.9109	0.9147

4 Conclusion

The paper described a framework for determining contextual polarity of ambiguous adjectives. The advantage of the proposed approach is that it does not rely on hand-crafted rules of opinion lexicons. Performance on a number of ambiguous adjectives is promising compared to a context-independent method using KLD. The proposed framework is extensible in a number of ways: features could be expanded to include, for instance, other dependency triples in the sentence or document, or on the contrary, filtered by the dependency relation type. Currently, we are working on various extensions of this framework, in particular, feature grouping, and are performing a larger scale evaluation on different corpora.

References

1. Esuli A. and Sebastiani F. Determining Term Subjectivity and Term Orientation for Opinion Mining. In Proc. of EACL, 2006.
2. Hu M. and Liu B. Mining and summarizing customer reviews. In Proc. of KDD, 2004.
3. Hatzivassiloglou, V. and McKeown, K. R. 1997. Predicting the semantic orientation of adjectives. In Proc. of ACL (pp. 174–181).
4. Wilson T., Wiebe J., Hoffman P. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In Proc. of EMNLP, 2005.
5. Popescu A. and Etzioni O. Extracting Product Features and Opinions from Reviews. In Proc. of EMNLP, 2005.
6. Ding X., Liu B. and Yu P. A holistic lexicon-based approach to opinion mining. In Proc. of WSDM'08.
7. Fahrni A. and Klenner M. Old Wine or Warm Beer: Target-specific Sentiment Analysis of Adjectives. In Proc. of the Symposium on Affective Language in Human and Machine, AISB 2008 Convention.
8. Spärck Jones K., Walker S., and Robertson S. E. 2000. A probabilistic model of information retrieval: Development and comparative experiments. Information Processing and Management, 36(6), 779–808 (Part 1); 809–840 (Part 2).
9. Vechtomova O. and Robertson S.E. 2012. A Domain-Independent Approach to Finding Related Entities. Information Processing and Management, 48(4), pp. 654-670.
10. Carpineto, C., De Mori, R., Romano, G., & Bigi, B. 2001. An information-theoretic approach to automatic query expansion. ACM ToIS, 19(1), 1–27.
11. Vechtomova O. 2010. Facet-based Opinion Retrieval from Blogs. Information Processing and Management, 46(1), 71-88.