Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 19-22 August 2007

EFFECTIVE ENTITY RESOLUTION IN PRODUCT REVIEW DOMAIN

JING-JING LIU[1], YUN-BO CAO[2], YA-LOU HUANG[1]

[1]Intelligent Information Processing Lab, Nankai University, Tianjin, China
[2]Microsoft Research Asia, Beijing, China
E-MAIL: nkicestone@gmail.com, Yunbo.Cao@microsoft.com, yellow@nankai.edu.cn

Abstract:

This paper is concerned with the problem of entity resolution in the product review domain. Specifically, given many references to product features, we would like to classify references related to one feature into a group. The product feature resolution is important to product review study, such as review ranking. To solve the problem, we propose an approach which combines two types of similarity characteristics: edit distance and context similarity. Experimental results indicate that the proposed approach resolves product features effectively and improves the performance of review ranking significantly.

Keywords:

Entity resolution; Product feature; Edit distance; Context similarity; Review ranking.

1. Introduction

Online shopping is a popular service on the World Wide Web nowadays. Many web sites such as Amazon.com and CNET.com are available for this purpose. These web sites provide an open communication platform where users can post reviews on products. Online shoppers can use these reviews as references when making purchase decisions. Manufacturers also monitor the reviews to investigate customer opinions and user requirements.

In research community, there exists an increasing interest in the study of product review recently. Kim et al. [15] and Zhang [16] try to rank product reviews to assess the quality of the reviews. Specifically, given a product name, e.g. 'Canon PowerShot SD600', all the reviews on this product will be ranked according to their quality. Meanwhile, in [6][7][8][13] the authors study the opinion summarization from reviews on the basis of product features. Here, product feature refers to the property of a product. For example, 'image quality' and 'auto focus' are two features of a digital camera. Positive and negative opinions from multiple reviews are aggregated on each feature, and the summarization on different features indicates the quality of the product in different aspects.

One problem in the existing methods is that reviews from different people may use different references to the same feature. For example, 'customer service' and 'support' are actually the same feature. Also, products of various brands may have different terminology. Therefore, before ranking and summarizing reviews, it is necessary to identify the different references to the same feature and merge them into a single one. Moreover, different web sites may keep different ontology for the specification of products (e.g. 'battery' in Amazon and 'power supply' in CNET). Therefore, when integrating reviews from different web sites or platforms (e.g. forums, blogs, news), it's important to build a uniform ontology of product features.

In this paper, we formulize the problem described above as an entity resolution problem in the domain of online product review. Thus, the problem is reduced to how to determine the equivalence of a product feature in different forms. We call this the 'resolution of product features'. Entity resolution has been studied extensively in the area of natural language processing [1][2][3][4][5]. We take the product feature as an entity and the various forms of the feature as references. The task is to map multiple references of the same entity into one group, i.e. to cluster the multiple references (e.g. 'battery' and 'power supply') into the same feature. We propose an approach which combines two similarity characteristics, and apply it to the review ranking problem. Our contributions in the paper are:

1) We address the entity resolution problem in the domain of online product review, namely the resolution of product features.
2) We exploit two methods of assessing similarity characteristics: edit distance and context similarity, and propose an approach to product feature resolution combining the two methods.
3) We integrate the resolution of product features into review ranking problem, and verify the effectiveness of the proposed approach with real world review data.

The rest of the paper is organized as follows. Section 2

1-4244-0973-X/07/$25.00 ©2007 IEEE

introduces the related work. Section 3 defines the problem. Section 4 explains the proposed approach to the resolution of product features. Section 5 reports the experimental results of the proposed approach and its application to review ranking problem. Section 6 summarizes our work in this paper and points out the future work.

2. Related Work

2.1. Entity Resolution

The entity resolution problem has been studied by many researchers. On et al. [5] propose an effective approach using quasi-clique. Kalashnikov and Mehrotra [4] use a probabilistic model for entity disambiguation using relationships. Bhattacharya and Getoor [3] employ a latent Dirichlet allocation model to revolve the entity resolution problem. Bontcheva et al. [2] propose a shallow method and Natrajan et al. [1] construct a semantic framework for entity resolution modeling. Methods of entity resolution vary among different domains. In this paper we address the entity resolution problem in the product review domain.

2.2. Extracting Product Features

The extraction of product features is proposed and studied by Liu et al [14]. The patterns of feature references are identified by the method of POS (Part-Of-speech) tagging and frequent patterns are learned from annotated data. These frequent patterns are taken as the candidates of product features. In [14], the matching of different product features is mentioned briefly and addressed by fuzzy matching. However, there exist many cases where the method fails to match the multiple references. In this paper we make a detailed yet systematic investigation of the multiple types of product feature resolution, and work on those cases that could not be easily resolved by fuzzy matching method.

2.3. Analysis of Online Reviews

Research work on the analysis of online reviews focuses on two areas. One is evaluating the helpfulness of reviews, and the other is opinion summarization. In [15][16], the evaluation of the helpfulness is formulized to a ranking problem and solved with a regression model. Features both on semantics and on syntax are employed in the learning model. As to the opinion summarization, several papers have been published [6][7][8][13]. Typically, in each sentence or a text segment of a review, the polarity of user's sentiment on a product feature is extracted. Then the aggregation of the polarities of individual sentiments on each product feature is presented as the summarization of user opinion. In this study, the performance of the summarization relies heavily on the resolution of product features, which are the basic units of the aggregation.

3. The Problem Definition

3.1. Extraction of Product Features

In this paper, we use the product reviews on digital cameras from Amazon.com as the data set. We crawled 23,914 reviews on 946 digital cameras from Amazon.com. From these reviews we use a method similar with [14] for the extraction of product features. We tag the POS of the reviews, extract noun words and phrases as candidates, and select those candidates with high frequency.

With this method we got 106 product feature candidates. We take this product feature set as a pool and analyze the problem of entity resolution in this pool. The following section analyzes the types of product feature resolution. Although these types are defined in the digital camera domain, they are independent of domain and can be adapted to other domains as well.

3.2. Types of Product Feature Resolution

- Type 1. Synonyms
  - *picture - photo - image*
  - *noise - speckle*
  - *manual – documentation*

  References in this type are single words with the same meaning in dictionary.
- Type 2. Different formats, same spelling
  - *auto focusing - autofocus*
  - *memory card - MemoryCard*
  - *point and shoot - point-n-shoot*

  References in this type are the phrases with the same spelling but in different formats.
- Type 3. Phrases with the same word
  - *LCD display - LCD screen*
  - *movie quality - video quality*
  - *power supply - power source*

  References in this type are the phrases that share one same word and have different other words.
- Type 4. Full string and sub string
  - *lens cap - cap*
  - *memory card - card*
  - *optical zoom - digital zoom – zoom*

  References in this type are pairs of references in which one is a sub string of the other.

- Type 5. Different orders of words
  - *quality of picture - picture quality*
  - *combination of features - feature combination*

  References in this type are the phrases that contain the same words yet in a different ordering.
- Type 6. Completely different words
  - *portable - easy to carry*
  - *out of focus - blurry*
  - *power supply – battery*

  References in this type are totally different words or phrases, which share no similarity in surface strings.

4. The Proposed Approach

In the natural language processing area, it's a general approach of taking WordNet[11] as a dictionary for semantic words matching. This method works well on synonyms (Type 1 in section 3.2). However, it cannot handle the cases such as 'battery life' and 'power'. In this paper, we leverage two other kinds of evidence to resolve the other types (from Type 2 to Type 6). One is surface string evidence; the other is contextual evidence. We use *edit distance* and *context similarity* to employ these two kinds of evidence, respectively.

4.1. Edit Distance

```
int EditDistance(char s[1..m], char t[1..n])
        // d is a table with m+1 rows and n+1
columns
    declare int d[0..m, 0..n]
    for i from 0 to m
        d[i, 0] := i
    for j from 1 to n
        d[0, j] := j
    for i from 1 to m
        for j from 1 to n
            if s[i] = t[j] then cost := 0
                else cost := 1
            d[i, j] := minimum(
                d[i-1, j] + 1,      // deletion
                d[i, j-1] + 1,      // insertion
                d[i-1, j-1] + cost  // substitution
                                )
  return d[m, n]
```

Figure 1. Algorithm for calculating edit distance

In the Types 2, 3, 4, the references in a pair share part of the same sub string. For these types of resolution, calculating the edit distance between the surface strings is an effective approach to learning the similarity between a pair of references.

The edit distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character [9][10].

For example, the edit distance between "rather" and "river" is 3, since these three edits change one into the other, and there is no way to do it with fewer than three edits:

1) rather → rither (substitution of 'a' for 'i')
2) rither → rivher (substitution of 't' for 'v')
3) rivher → river (delete 'h')

4.2. Context Similarity

In resolution Types 5 and 6, the references in a pair differ significantly in surface string; thus they cannot be resolved with the method of calculating edit distance. And in Types 3 vs. 4, only using the edit distance will bring much noise to the resolution; therefore the edit distance fails to resolve all the cases in these types either. To improve the performance of resolution, we also use context similarity which is effective evidence in these cases.

Context similarity is calculated based on Vector Space Model (VSM [18]). VSM is an algebraic model representing documents in a formal manner through the use of vectors of identifiers (e.g. index terms) in a multi-dimensional linear space. The set of terms is a predefined collection of terms, for example, the set of all unique words occurring in the document corpus.

In information retrieval, relevancy rankings of documents in a keyword search can be calculated, by comparing the deviation of angles between each document vector and the original query vector where the query is represented as the same kind of vector as the documents. The equation of calculating the similarity is given by:

$$\cos\theta = \frac{\mathbf{v}_1 \bullet \mathbf{v}_2}{\|\mathbf{v}_1\| \bullet \|\mathbf{v}_2\|} \quad (1)$$

Here we borrow the document similarity theory, and use the similarity between the contexts around a pair of references to compare their similarity.

4.3. The Proposed Approach

In this paper, we propose an effective approach to the problem of product feature resolution. The approach combines two kinds of similarity characteristics: edit distance and content similarity. The algorithm is as follows:

In the method of using edit distance, the equation of calculating the similarity score of two references is given by:

$$EDScore(R_1, R_2) = \frac{EditDistance(R_1, R_2)}{\frac{1}{2}(Len(R_1) + Len(R_2))} \quad (2)$$

where R_1, R_2 represent two references, respectively. $EDScore(R_1, R_2)$ represents the *Edit-Distance-Score* of a pair of references. $EditDistance(R_1, R_2)$ represents the edit distance between the surface strings of two references (minimum number of operations needed to transform one string into the other). $Len(R)$ represents the length of string in reference R. The higher *EDScore* a pair of references get, the lower similarity the two references have.

In the method of using context similarity, we split all the reviews into sentences. For each reference, we take it as a query and search for all the relevant sentences to the query. Then we build a bag-of-words with all the terms occur in the relevant sentences. We then construct a vector for the reference, by taking each unique term in the bag-of-words as a dimension of the vector. The equation of calculating the similarity between each pair of vectors is given by:

$$CSScore(R_1, R_2) = \left| \frac{\mathbf{v}(R_1) \bullet \mathbf{v}(R_2)}{\|\mathbf{v}(R_1)\| \bullet \|\mathbf{v}(R_2)\|} \right| \quad (3)$$

where $\mathbf{v}(R)$ represents the vector of bag-of-words of R. $CSScore(R_1, R_2)$ represents the *Context-Similarity-Score* of a pair of references. The higher *CSScore* a pair of references get, the higher similarity the two references have.

We set a threshold for each method in order to filter the pairs with low similarity. Only the pairs whose *EDScore* scores are lower than the threshold of *EDScore* or those whose *CSScore* scores are higher than the threshold of *CSScore* will be taken as similar pairs.

Then we define a mixed score as a combination of the two methods. The equation of calculating the mixed score is given by:

$$MixScore(R_1, R_2) = \lambda * N_f(CSScore(R_1, R_2)) * f(R_1 + (1-\lambda) * N_g(EDScore(R_1, R_2)) * g(R_1, R_2, T_g) \quad (4)$$

where $MixScore(R_1, R_2)$ represents the *Mixed-Score* of *Edit-Distance-Score* and *Context-Similarity-Score*. λ represents the parameter that controls the weights of each score. $f(R_1, R_2)$ represents the indicator function of *CSScore*. If the *CSScore* is higher than the given threshold T_f, $f(R_1, R_2)$ =1; else $f(R_1, R_2)$ =0. Similarly, $g(R_1, R_2)$ represents the indicator function of *EDScore*. If the *EDScore* is lower than the given threshold Tg, $g(R_1, R_2)$ =1; else $g(R_1, R_2)$ =0. $N_f(CSScore)$ represents the normalizing function for *CSScore* while $N_g(EDScore)$ represents the normalizing function for *EDScore*. These two normalizing functions aim to smooth the two scores into the same scale for a finer combination.

4.4. Review Ranking

The problem of review ranking is to assess the quality of different reviews on one product [15][16]. Given a query of product name, e.g. 'Canon PowerShot SD600', all the reviews on this product will be sorted by the quality of each review. The higher a review is ranked, the higher quality the review has.

A more formal definition of the problem is as follows: given a training data set, we construct a model that can minimize error in prediction of *y* given *x* (generalization error). Here *x* represents a review and *y* represents a score which reflects the ranking position of *x*. When applied to a new instance *x*, the model predicts the corresponding *y* and outputs the score of the prediction.

The formula of the ranking algorithm is give by:

$$f(x) = w^T x + b \quad (5)$$

where $\boldsymbol{x}$ refers to the feature vector of review employed in the learning process; $\boldsymbol{w}$ represents the vector of weights for each feature; and *b* denotes an intercept. ***f(x)*** refers to the score of the ranking result. The higher the value of ***f(x)*** is, the higher the quality of the instance $\boldsymbol{x}$ is.

We implement the review ranking problem with this supervised learning approach. In our experiments, we utilize the SVM Light toolkit [17] as the learning model. Details of the learning algorithm can be found in [12]. Briefly, the learning algorithm creates the 'hyper plane' in equation (5), such that the hyper plane separates the positive and negative instances in the training data with the largest 'margin'. Part of the features employed in the learning model for ranking are listed in Table 1.

Table 1. Part of features employed in ranking model

Length of review
TF-IDF
Number of unique product features mentioned
Frequency of each mentioned product feature
Number of mentioned features in title of review
Number of mentioned products
Number of mentioned products in title of review
Percentage of sentences with sentiment (positive & negative) in all sentences
A rating the reviewer gives to the product
Number of comparative references (better, worse, etc.)
Number of paragraphs
Average length of paragraphs

5. Experiments

5.1. Resolution of Product Features

We crawled 23,914 reviews on 946 digital cameras from Amazon.com. From these reviews we extracted 106 product features on digital camera as candidates, as mentioned in section 3.1. We manually labeled resolution pairs from these 106 candidates and used this annotation as the ground truth for the resolution of product features. We compared the experimental results of the proposed approach with this ground truth to see how many pairs have been found correctly, as shown in Table 2.

We used recall and precision as the evaluation methods. Recall is the fraction of all correct answers that is returned by the search. Precision is the percentage of returned answers that are correct.

$$Recall = \frac{|CorrectAnswersFound|}{|CorrectAnswers|} \quad (6)$$

$$Precision = \frac{|CorrectAnswersFound|}{|AnswersFound|} \quad (7)$$

As for edit distance, we used the surface strings of the reference candidates to calculate the similarity between each pair of references, as shown in equation (2). As for context similarity, we used the 23,914 reviews as the context corpus. We split the reviews into sentences. Given each reference, we retrieved all the sentences that are relevant to the reference, independent of reviews. Then we used all the terms around the reference in these sentences as the bag-of-words for this reference. In this way we built a vector of context for each reference. The context similarity was then calculated based on these vectors with equation (3). At last, we combined the two scores with the equation (4).

Table 2 shows the experimental results of each method respectively. We could see that as for the method using edit distance, the precision drops when the threshold increases, while the recall decreases. As for the context similarity method, it's on the opposite. We prefer to obtain high precision in the expense of recall, while keeping a relatively high recall to maintain the amount of candidate answers. The cost of precision in the relatively high recall will be made up by the combination of the two methods. Therefore, we chose the threshold of 0.5 as the local parameter for edit distance, and the threshold of 0.6 as the local parameter for context similarity. We tried different settings for λ in equation (4) and selected the setting of 0.5 empirically as the global parameter.

Under these settings, we got a collection of resolution pairs, which we used in the experiments on review ranking in the following section.

Table 2. Experimental results of resolution

	Threshold	Recall	Precision
Threshold for Edit Distance	0.4	0.26	1
	0.5	0.39	0.8
	0.6	0.45	0.58
	0.7	0.45	0.47
	Threshold	Recall	Precision
Threshold for Context Similarity	0.8	0.46	0.86
	0.7	0.46	0.86
	0.6	0.52	0.87
	0.5	0.54	0.45
	0.4	0.54	0.13

5.2. Product Review Ranking

We used the 23,914 reviews on 946 digital cameras as the data set for review ranking. We sorted the 946 digital cameras by the number of reviews they held, as shown in Figure 2, and selected the top 100 cameras which held more than 90 reviews. We then randomly divided the 100 cameras into 50 training queries and 50 test queries.

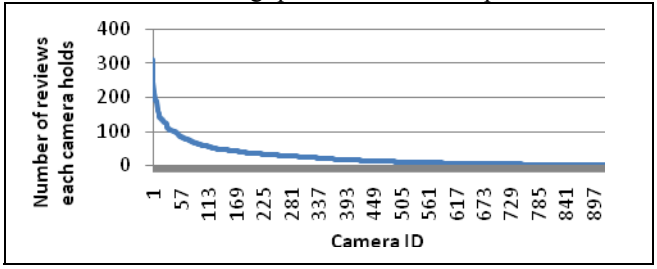

Figure 2. Distribution of reviews on cameras

We took the collection of resolution pairs from the experiments in section 5.1 as the resolution dictionary. Then we applied product feature resolution to review ranking. That is, when generating features for learning, e.g. calculating how many different product features a review mentioned, entity resolution was involved and the pairs of similar references were mapped to a unique product feature. For example, in a review containing 3 occurrences of 'image quality' and 2 occurrences of 'quality of picture', the number of product features would be 1, and the frequency of this unique product feature would be 5. Pay attention that, when several pairs shared a same reference, the references in these pairs were all mapped to a unique product feature. In this way we clustered the reference pairs into different feature groups.

We employed SVM light [17] as the ranking model and used the features listed in section 4.4 to train the learning model. We trained the model with the 50 training queries and tested it on the 50 test queries. The ground truth of review ranking was based on an annotation on the

quality of the reviews on these 100 cameras.

For the evaluation of review ranking, we used NDCG (Normalized Discounted Cumulative Gain) as the measure. NDCG is a measure commonly used in IR.

$$NDCG \propto \sum_{i} \frac{2^{rel(i)} - 1}{\log(1+i)} \quad (8)$$

We tried two runs of review ranking, one involved the resolution of product features and the other didn't. Figure 3 shows the experimental results. The higher bar is the result involving the resolution of product features, the lower bar is the result without the consideration of product feature resolution. From this figure we can see that the NDCG of review ranking from @1 to @15 all rise when involving the resolution of product features. This proves that the resolution is very helpful to review ranking, and can improve the performance of ranking significantly.

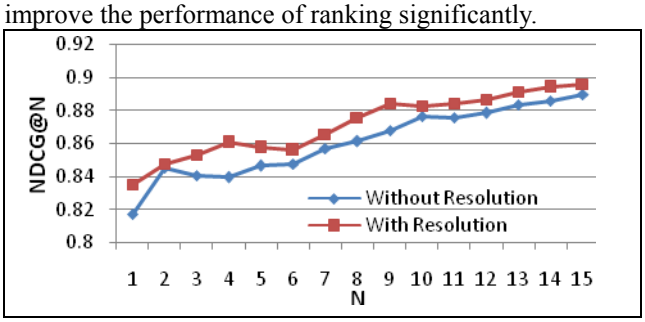

Figure 3. Experimental results of review ranking

6. Conclusions & Future Work

In this paper we propose a combination approach to the problem of entity resolution in product review domain. Methods of employing two types of similarity characteristics: edit distance and context similarity are investigated. An effective combination of these two methods is implemented. Experimental results based on real world dataset indicate that the proposed approach to product feature resolution significantly improve the performance of review ranking task. As for the future work, we would like to focus on the following two aspects: (1) building bag-of-words on larger corpus in the method of using context similarity in order to remove the bias of local contexts; (2) exploring other combination methods such as non-linear models.

References

[1] Anand Natrajan, Paul F. Reynolds Jr, Sudhir Srinivasan, MRE: A Flexible Approach to Multi-Resolution Modeling

[2] Kalina Bontcheva, Marin Dimitrov, Diana Maynard, Valentin Tablan, Hamish Cunningham, Shallow Methods for Named Entity Coreference Resolution

[3] Indrajit Bhattacharya, Lise Getoor, A Latent Dirichlet Allocation Model for Entity Resolution

[4] Dmitri V. Kalashnikov and Sharad Mehrotra, A probabilistic model for entity disambiguation using relationships, TR-RESCUE-04-12

[5] B. On, E. E, D. Lee, J K, J Pei, An effective approach to entity resolution problem using quasi-clique and its application to digital libraries, JCDL'06

[6] AM Popescu and O Etzioni. Extracting product features and opinions from reviews. HLT-EMNLP'05.

[7] B. Liu, M. Hu, J. Cheng. Opinion observer: analyzing and comparing opinions on the web. WWW '05.

[8] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. ACL'04.

[9] Dekang. Lin. Automatic retrieval and clustering of similar words. COLING-ACL'98.

[10] Esko Ukkonen. Algorithms for approximate string matching. Information and Control. 1985.

[11] C. Fellbaum. WordNet: an Electronic Lexical Database, MIT Press. 1998.

[12] Harris Drucker, Chris J.C., Burges Linda Kaufman, Alex Smola and Vladimir Vapnik. Support vector regression machines. Advances in Neural Information Processing Systems. 1997.

[13] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. WWW'03.

[14] Minqing. Hu and Bing. Liu. Mining Opinion Features in Customer Reviews. AAAI'04.

[15] Soo-Min Kim, Patrick Pantel, Tim Chklovski, Marco Pennacchiotti. Automatically Assessing Review Helpfulness. EMNLP'06.

[16] Zhu Zhang and Balaji Varadarajan. Utility Scoring of Product Reviews. CIKM'06

[17] Thorsten Joachims. SVMlight -- Support Vector Machine. http://svmlight.joachims.org/, 2004.

[18] G. Salton, A. Wong, and C. S. Yang, A Vector Space Model for Automatic Indexing, Communications of the ACM. 1975