Entity Set Expansion in Opinion Documents

Lei Zhang and Bing Liu
Department of Computer Science
University of Illinois at Chicago
851 S. Morgan St., Chicago, IL 60607
{lzhang3, liub@cs.uic.edu}

ABSTRACT

Opinion mining has been an active research area in recent years. The task is to extract opinions expressed on entities and their attributes. For example, the sentence, "*I love the picture quality of Sony cameras*," expresses a positive opinion on the *picture quality* attribute of *Sony* cameras. *Sony* is the entity. This paper focuses on mining entities (e.g., *Sony*). This is an important problem because without knowing the entity, the extracted opinion is of little use. The problem is similar to the classic *named entity recognition* problem. However, there is a major difference. In a typical opinion mining application, the user wants to find opinions on some competing entities, e.g., competing or relevant products. However, he/she often can only provide a few names as there are too many of them. The system has to find the rest from a corpus. This implies that the discovered entities must be of the same type/class. This is the set expansion problem. Classic methods for solving the problem are based on distributional similarity. However, we found this method is inaccurate. We then employ a learning-based method called *Bayesian Sets*. However, directly applying *Bayesian Sets* produces poor results. We then propose a more sophisticated way to use *Bayesian Sets*. This method, however, causes two major problems: entity ranking and feature sparseness. For entity ranking, we propose a re-ranking method to solve the problem. For feature sparseness, we propose two methods to re-weight features and to determine the quality of features. These methods help improve the mining results substantially. Additionally, like any learning algorithm, Bayesian Sets requires the user to engineer a set of features. We design some generic features based on part-of-speech tags of words for learning, which thus does not need to engineer features for each specific domain. Experimental results using 10 real-life datasets from diverse domains demonstrated the effectiveness of the proposed technique.

Categories and Subject Descriptors

H.3.1[Information Storage and Retrieval]: Content Analysis and Indexing

General Terms

Algorithms, experimentation

Keywords

Entity extraction, set expansion, text mining

1. INTRODUCTION

In recent years, opinion mining or sentiment analysis has been an active research area in both NLP and Web mining. The basic task is to extract people's opinions expressed on entities and attributes of the entities. The entities are usually brand or product names, and organization names. For example, the sentence, "*I brought a Sony camera yesterday, and its picture quality is great*," expresses a positive opinion on the *picture quality* attribute of the *Sony* camera. Here *Sony* is the entity. Opinion classification (determining whether an opinion is positive or negative) and attribute discovery have been studied many by researchers [20][24]. In this work, we focus on identifying entities (e.g., *Sony* in our example) in opinion documents. This is an important problem because without knowing the entity that an opinion has been expressed on, any extracted attributes or opinions are of little use. For instance, if we only know that someone expressed a positive opinion on picture quality, but do not know on which camera, then the piece of opinion has little value.

The problem of discovering entities is similar to the traditional named entity recognition problem. However, there is a major difference. In a typical opinion mining application, the user wants to find opinions on some competing entities, e.g., competing products or brands (e.g., Canon, Sony and many more). However, the user often can only provide a few names because there are so many different brands and models. Web users also write names of the products in various ways in forums and blogs. It is thus important for a system to automatically discover them from relevant corpora (e.g., blogs and forum discussions about cameras). The key requirement of this discovery is that the discovered entities must be relevant, i.e., they must be of the same class/type as the user provided entities. In the example above, the system should only discover camera brands and models.

This problem is the set expansion problem [12][29][22][30], which expands a set of given seed entities. Formally, the problem is stated as follows: Given a set Q of seed entities of a particular class C, and a set D of candidate entities, we wish to determine which of the entities in D belong to C. That is, we "grow" the class C based on the set of seed examples Q. Clearly, this is a classification problem which needs a binary decision for each entity in D (belonging to C or not belonging to C). However, in practice, the problem is often solved as a ranking problem, i.e., to rank the entities in D based on their likelihoods of belonging to C. In our context, the user-given entities are the set of initial seeds. The system needs to expand the set using a text corpus.

The classic methods for solving the problem are based on distributional similarity [17][19][22][23][26].This approach works by comparing the similarity of the word distribution of the surround words of a candidate entity and the seed entities, and then ranking the candidate entities based on the similarity values. However, our results show that this approach is inaccurate. In this paper, we employ a machine learning approach called *Bayesian*

Sets [11]. The method estimates the probability that each candidate belongs to a "hidden" class represented by the seeds. It then produces a ranking of the candidates. The data set required is in the usual form for learning. Each example is a feature vector.

This technique can be directly applied to our context as follows. Given a set of seed entities and a text corpus, the seed data, a set of feature vectors (denoted by *R*) can be produced as follows:

1. A set of features is first designed to represent each entity.
2. For each seed entity, we identify all the sentences in the corpus that contain the entity. Based on the sentence contexts, we produce a feature vector to represent the seed entity.

Candidate entities and their feature vectors (denoted by *T*) can be generated from the corpus similarly:

1. From the corpus, we identify all candidate entities and sentences that contain them. We will discuss how this can be done later in Section 4.1.
2. For each candidate entity, a feature vector is produced based on all the sentences that contain the candidate entity.

The sets *R* and *T* are then used for Bayesian Sets learning. The learned model then assign a relevance score to each test vector representing a candidate entity. The score is then used to rank the candidate entities. The ranking is the final result.

Unfortunately, this direct application of Bayesian Sets produces poor results. We believe there are two main reasons. First, since Bayesian Sets uses binary features, multiple occurrences of an entity in the corpus, which give rich contextual information, is not fully exploited. Second, since the number of seeds is very small, the learned results from Bayesian Sets can be quite unreliable.

We propose a more sophisticated method to use Bayesian Sets, which produces much better results. We introduce it here. Given a set of seed entities, the seed data are produced as follows:

1. Again, a set of features to represent each entity is designed.
2. For each seed entity, we identify all sentences in the corpus that contain the entity. From each sentence, a separate feature vector representing the seed entity in the sentence is produced. Hence, for each seed entity, we produce multiple vectors in the seed data. The number of vectors for the entity is the same as the number of sentences containing the entity.

Candidate entities and their data can be generated similarly:

1. From the corpus, all the candidate entities and sentences containing them are first identified.
2. For each candidate entity **e**, and for each sentence *s* that contains the entity, a feature vector is produced based on sentence *s* for entity **e**. Again, each candidate entity generates multiple feature vectors in the candidate data.

Clearly, the difference between this approach and the direct approach is that in the direct approach, each entity generates a single vector, while in the new approach each entity generates multiple vectors depending on the number of sentences containing the entity. Based on the generated data, we can run Bayesian Sets, and rank the feature vectors of the candidate entities.

This approach, however, causes two problems. The first problem is that since each candidate entity has multiple vectors and each vector has a different score produced by Bayesian Sets, the question is which score to use to rank the candidate entity. We need to decide this because the final result that we want is a ranked list of entities, not the vectors. This problem does not exist in the direct approach.

Before we deal with this problem, let us also describe an important issue that has not been addressed in the set expansion literature. The issue is the entity occurrence frequency. We consider an entity e_1 is more important than another entity e_2 if e_1 appears more frequently than entity e_2. In practice, it is desirable to rank those frequent entities higher than infrequent entities. The reason is that missing a frequently mentioned entity in opinion mining is very bad, but missing a rare entity is not a big issue. To deal with the two issues, we propose an effective method to combine both factors to rank entities with very good results.

The second problem is the feature sparseness. Due to the fact that each vector is only based on a single sentence that contains a candidate entity, the number of features contained in a sentence can be very small and even 0. Making best use of each feature becomes crucial since the score function learned from Bayesian Sets is essentially a weight vector corresponding to all features. Here again, we developed two techniques to remedy this situation. One of them is based on feature scaling and the other is based on automatically determining the quality of each feature, and then uses the feature quality information to re-weight each feature.

As with any supervised learning approach, for each application the user needs to design a new set of features. This paper proposes a set of generic features for learning which have been shown to perform very well in diverse domains. This means that the user does not need to engineer features for each application.

Two more improvements were also made by enlarging the user-given seeds by using coordination patterns and by bootstrapping Bayesian Sets. All the proposed methods have been implemented and tested based on 10 real-life data sets (or text corpora) used for opinion mining. The data sets were collected from different review sites, user discussion forums and blogs on the Web. They have been used by a commercial company that provides opinion mining services for their clients. In the evaluation, we compare the new approach with a popular distribution similarity method, and a semi-supervised learning method. Our experimental results show that the new method outperforms them by a large margin.

2. RELATED WORK

The proposed research is in the general area of information extraction, and more specifically named entity recognition (NER). There are extensive literatures on the topic, which can be grouped into three main categories, supervised learning, unsupervised learning and semi-supervised learning.

Supervised approaches are currently the dominant technique for solving the NER problem. Conditional Random Fields (CRF) is perhaps the most effective method [16][21]. These methods need a large collection of labeled training examples. Labeling data is a labor-intensive and time-consuming process. Labeling also needs to be done for each domain. It is thus not suitable for applications that involve a large number of domains such as opinion mining. Our problem is also different as we have only a small number of seed entities. It is thus more related to semi-supervised learning.

Unsupervised approaches gather named entities from clustered groups based on the similarity of their contexts. This method usually relies on external knowledge (e,g., WordNet and Wikipedia pages [3]), class-specific extraction patterns [3][7], corpus-based term similarity [23] and n-gram collocations [9] to find term clusters. Our work does not use the unsupervised method because we need a specific class of entities that are similar to the seeds provided by the user.

In semi-supervised approach, we have a set of seed entities. The system uses either class-specific patterns to populate entity term class or distributional similarity to find terms similar to the terms in the seed set [27][1][3][9][19][22][23][25][31][32].

Distributional similarity is the state-of-the-art technique for solving this problem. There are many distributional similarity measures [19][2] (see [17] for a survey and evaluation of these methods). We will show that this approach does not perform well for our data sets. It is able to find only those very frequently mentioned entities as they have more contextual information, but not those less frequently mentioned entities due to their weaker contextual information. [18] adopts a machine learning approach called *positive and unlabeled learning* (PU learning) to model set expansion problem. PU learning is a semi-supervised learning model, which learns from positive and unlabeled examples. Bayesian Sets [11][14] can also be seen as a semi-supervised method, which is based on Bayesian inference. We will discuss it in detail in the next section. It is worth mentioning that [18] also compares performance between distributional similarity, Bayesian Sets and PU learning method. It shows that PU learning method (S-EM) outperformed the other two methods. However, feature engineering and feature sparseness problem were neglected for Bayesian Sets. We discovered that if we can exploit the proposed techniques in this paper, e.g., applying feature engineering, tackling feature sparseness, and enlarging seed sets, Bayesian Sets can achieve better result than the PU Learning method.

On the Web, Google Sets is perhaps the first system, which expands a seed set by detecting lists of items based on regularities in the HTML code [12]. It assumes that the items in a list belong to the same class. The Boo!Wa! system [30] uses a similar idea and is based on Web wrapper technologies to extract and rank entities iteratively.

3. INTRODUCTION TO BAYESIAN SETS

Let D be a collection of items and Q be a user-given *seed set* of items, which is a (small) subset of D (i.e., $Q \subseteq D$). The task of Bayesian Sets is to use a model-based probabilistic criterion to give a score to each item **e** in D ($\mathbf{e} \in D$) to gauge how well **e** fits into Q. In other words, it measures how likely **e** belongs to the "*hidden*" class represented/implied by Q. Each item **e** is represented with a binary feature vector.

The Bayesian criterion score for item **e** is expressed as follows:

$$score(\mathbf{e}) = \frac{p(\mathbf{e}|Q)}{p(\mathbf{e})} \tag{1}$$

$p(\mathbf{e}|Q)$ represents how probable that **e** belongs to the same class as Q given the examples in Q. $p(\mathbf{e})$ is the prior probability of item **e**. Using Bayes rule, the equation can be re-written as:

$$score(\mathbf{e}) = \frac{p(\mathbf{e},Q)}{P(\mathbf{e})P(Q)} \tag{2}$$

Eqn. (2) can be interpreted as the ratio of the joint probability of observing **e** and Q, to the probability of independently observing **e** and Q. The ratio basically compares the probability that **e** and Q are generated by the same model with parameters θ, and the probability that **e** and Q are generated by different models with different parameters θ and $\widetilde{\theta}$ (Fig. 1). Eqn. (2) says that if the probability that **e** and Q are generated from the same model with the parameters θ is high, the score of **e** will be high. On the other hand, if the probability that **e** and Q come from different models with different parameters θ and $\tilde{\theta}$ is high, the score will be low.

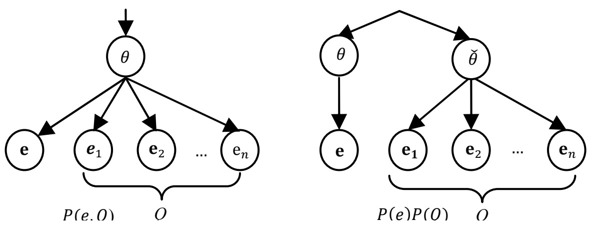

Fig. 1. Two hypotheses that **e** and Q are generated

In pseudo code, the Bayesian Sets algorithm is given in Fig. 2.

```
Algorithm: BayesianSets(Q, D)
  Input: A small seed set Q of entities
         A set of candidate entities D (= {e_1 , e_2 , e_3 ... e_n})
  Output: A ranked list of entities in D
1.  for each entity e_i in D
2.        compute: $score(\mathbf{e}_i) = \frac{p(\mathbf{e}_i,Q)}{p(\mathbf{e}_i)p(Q)}$
3.  end for
4.  Rank the items in D based on their scores;
```

Fig. 2. The Bayesian Sets learning algorithm

If we assume that $\mathbf{q}_k \in Q$ is independently and identically distributed (i.i.d.) and Q and $\mathbf{e}_i$ come from the same model with the same parameters θ, each of the three terms in Eqn. (2) are marginal likelihoods and can be written as integrals of the following forms:

$$p(Q) = \int \left[\prod_{\mathbf{q}_k \in Q} p(\mathbf{q}_k|\theta)\right] p(\theta) d\theta \tag{3}$$
$$p(e_i) = \int p(\mathbf{e}_i|\theta) p(\theta) d\theta \tag{4}$$
$$p(e_i, Q) = \int \left[\prod_{\mathbf{q}_k \in Q} P(\mathbf{q}_k|\theta)\right] p(\mathbf{e}_i|\theta) p(\theta) d\theta \tag{5}$$

Let us first compute the integrals of Eqn. (3). Each seed entity $\mathbf{q}_k \in Q$ is represented as a binary feature vector (q_{k1}, q_{k2} , ... q_{kJ}). We assume each element of the feature vector has an independent Bernoulli distribution:

$$p(\mathbf{q}_k|\theta) = \prod_{j=1}^{J} \theta_j^{q_{kj}} (1-\theta_j)^{1-q_{kj}} \tag{6}$$

The conjugate prior for the parameters of a Bernoulli distribution is the Beta distribution:

$$p(\theta|\alpha,\beta) = \prod_{j=1}^{J} \frac{\Gamma(\alpha_j+\beta_j)}{\Gamma(\alpha_j)\Gamma(\beta_j)} \theta_j^{\alpha_j-1} (1-\theta_j)^{\beta_j-1} \tag{7}$$

where α and β are hyperparameters (which are also vectors). We set α and β empirically from the data, $\alpha_j = km_j$, $\beta_j = k(1- m_j)$, where m_j is the mean value of jth components of all possible entities, and k is a scaling factor (we use 1 in our experiments and it works well). The Gamma function is a generalization of the factorial function.

For $Q = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_N\}$, Eqn. (3) can be represented as follows:

$$p(Q|\alpha,\beta) = \prod_j \frac{\Gamma(\alpha_j+\beta_j)}{\Gamma(\alpha_j)\Gamma(\beta_j)} \frac{\Gamma(\tilde{\alpha}_j)\Gamma(\tilde{\beta}_j)}{\Gamma(\tilde{\alpha}_j+\tilde{\beta}_j)} \tag{8}$$

where $\tilde{\alpha}_j = \alpha_j + \sum_{k=1}^{N} q_{kj}$ and $\tilde{\beta}_j = \beta_j + N - \sum_{k=1}^{N} q_{kj}$. With the same idea, we can compute Eqn. (4) and Eqn. (5).

Overall, the score of $\mathbf{e}_i$, which is also represented a feature vector, (e_{i1}, e_{i2} , ... e_{iJ}), is computed with:

$$score(\mathbf{e}_i) = \prod_j \frac{\frac{\Gamma(\alpha_j+\beta_j+N)}{\Gamma(\alpha_j+\beta_j+N+1)} \frac{\Gamma(\tilde{\alpha}_j+e_{ij})\Gamma(\tilde{\beta}_j+1-e_{ij})}{\Gamma(\tilde{\alpha}_j)\Gamma(\tilde{\beta}_j)}}{\frac{\Gamma(\alpha_j+\beta_j)}{\Gamma(\alpha_j+\beta_j+1)} \frac{\Gamma(\alpha_j+e_{ij})\Gamma(\beta_j+1-e_{ij})}{\Gamma(\alpha_j)\Gamma(\beta_j)}} \tag{9}$$

We can simplify Eqn. (9) by using the fact that $\Gamma(x) = (x-1)\Gamma(x-1)$ for $x > 1$. In addition, for each j we can consider the two cases $e_{ij} = 0$ and $e_{ij} = 1$ separately. For $e_{ij} = 0$, we have a contribution $\frac{\alpha_j+\beta_j}{\alpha_j+\beta_j+N}\frac{\tilde{\beta}_j}{\beta_j}$. For $e_{ij} = 1$, we have a contribution $\frac{\alpha_j+\beta_j}{\alpha_j+\beta_j+N}\frac{\tilde{\alpha}_j}{\alpha_j}$.

Putting above together, we have Eqn. (10).

$$score(\mathbf{e}_i) = \prod_j \frac{\alpha_j+\beta_j}{\alpha_j+\beta_j+N}\left(\frac{\tilde{\alpha}_j}{\alpha_j}\right)^{e_{ij}}\left(\frac{\tilde{\beta}_j}{\beta_j}\right)^{1-e_{ij}} \quad (10)$$

The log of the score is linear in $\mathbf{e}_i$:

$$\log\, score(\mathbf{e}_i) = c + \sum_j w_j e_{ij} \quad (11)$$

where

$$c = \sum_j \log(\alpha_j+\beta_j) - \log(\alpha_j+\beta_j+N) + log\tilde{\beta}_j - log\beta_j$$

and $w_j = \log\tilde{\alpha}_j - log\alpha_j - log\tilde{\beta}_j + log\beta_j$ (12)

All possible entities $\mathbf{e}_i$ will be assigned a similarity score by Eqn. (11). Then we can rank them accordingly. The top ranked entities should be highly related to the seed set Q according to the Bayesian Set algorithm.

4. CANDIDATE ENTITIES AND ENTITY FEATURES

In this section, we discuss how to generate candidate entities, i.e., the set *D*, and design features for Bayesian Sets learning. A template-based feature identification technique is proposed which requires no manual engineering of individual features. We also discuss the new method of data (feature vector) generation.

4.1 Candidate Entity Extraction

Since we are interested in named entities, we select single words and phrases as candidate entities based on their corresponding part-of-speech (POS) tags. A word is an entity candidate e ($e \in D$) if its corresponding part-of-speech (POS) tag is NNP (proper noun), NNPS (plural proper noun), or CD (cardinal number). NNP and NNPS are normally the POS tags of words starting with an uppercase letter (which include words with every letter in uppercase). We regard a phrase with a sequence of NNP, NNPS and CD POS tags as one candidate entity (CD cannot be the first word unless it starts with a letter), e.g., "Windows/NNP 7/CD" and "Nokia/NNP N97/CD" are regarded as two candidates "Windows 7" and "Nokia N97".

4.2 Template-Based Feature Identification

As noted above, each candidate entity in Bayesian Sets is represented as a binary feature vector. For an entity, if a feature value is 1, it means that the entity has the feature and 0 otherwise.

Like a typical learning algorithm, one has to design a set of features for learning. Our feature set consists of two subsets:

1. **Entity word features (EWF)**: These features characterize the words representing entity themselves. This set of features is completely domain independent as we will see below.
2. **Surrounding word features (SWF)**: These features are the surrounding words of a candidate entity.

Entity word features are about the makeup and characteristics of a word. They describe different word case combinations, and digits in words. We use the following word features:

EWF1: Only the first letter in the word is capitalized, e.g., Sony.
EWF2: Every letter in the word is capitalized, e.g., IBM.
EWF3: There is at least one letter in the word that is capitalized and it is not the first letter, e.g., iPhone.
EWF4: The word has a combination of letters and digits, e.g., N97.
EWF5: The word consists of only digits.

For an entity which is a phrase, as long as any word has a feature above, that feature value is set to 1.

Regular expressions are used to detect the word features. Note that in this work we are interested in English text. We are aware that some features may not work well in other languages. For other languages, these features may need to be revised.

Surrounding word features are words on the left or right of the candidate entity. So we define the following six syntactic templates for feature extraction. The extracted features are specific words or phrases. In the templates below, EN refers to an entity/candidate. We use 5 words before and 5 words after the entity phrase/word as the text window. This window size works very well (but it can be changed).

Template 1: Left first verb of EN in the text window
For example, in the sentence, "*I have bought this EN LCD yesterday*", "bought" is extracted as a feature as it is the first verb on the left of EN.

Template 2: Left first noun of EN in the text window
For example, in the sentence "*I have taken 10 mg of EN*", "mg" is extracted as a feature as it is the first noun on the left of EN.

Template 3: Left first noun-preposition phrase of EN in the text window
For example, in the sentence, "*The performance of EN is awesome*", the phrase "performance of" is extracted as a feature as it is the first noun-preposition phrase on the left of EN. Note that the phrase is used as a feature.

Template 4: Right first verb of EN in the text window
For example, in the sentence "*This EN works*", "works" is extracted as a feature as it is the first verb on the right of the entity EN.

Template 5: Right first noun of EN in the text window
For example, in the sentence "*I went to a EN dealer*"; "dealer" is extracted as a feature as it is the first noun on the right of EN.

Template 6: Right first preposition-noun phrase of EN in the text window
For example, in the sentence, "*I am using EN for Arthritis*"; "for Arthritis" is extracted as a feature as it is the first preposition-noun phrase on the right of EN.

Note that noun here includes all forms of nouns, and verb here also includes all forms of verbs, they are determined by their POS tags. Stemming is also applied to reduce words to their stems.

For feature extraction, we only use the seed entities and the sentences in the given corpus that contain the seed entities. More specifically, we perform the following steps:

1. Identify all sentences that contain seed entities in the corpus.
2. For each sentence and for each entity, we use the above templates to match the left and the right contexts of the entity within the text window and extract all the matched words. These matched words are the pattern features.

Our experiments show the above templates perform very well in a wide range of domains, which is not surprising because the templates are so general and comprehensive that they basically cover all kinds of nearby context words except adjectives and adverbs. Adjective and adverbs are often non-specific. Any such word can modify almost anything, while in a domain some specific verbs and nouns are often associated with entities. Even we have this set of highly general templates, by no means, we claim/prove that these templates are sufficient for entity extraction in any domain. The approach of using templates, however, makes feature engineering much easier, compared to manually identifying words or phrases in each new domain.

4.3 Data Generation

Based on the seed entities Q and candidate entities D, and the extracted features, we can prepare data for learning. As we discussed in the introduction section, we do not use the conventional way to generate the data, which generates one vector for each entity in the seed set or the candidate entity set identified in the corpus, as it does not work well. Instead, for each appearance of an entity in a sentence, a feature vector is generated. The process has been discussed in the introduction section, and we will not repeat it here. Of course, this way of preparing the data causes the problems of feature sparseness and entity ranking, which we solve in Sections 5 and 6.

5. FEATURE REWEIGHTING

As we discussed in the introduction section, the way that we prepare the data causes serious data sparseness because each vector is only obtained from one sentence (either long or short) and because the features are only from sentences containing the given seeds. The number of pattern features in a vector can be very small or even 0. This requires best use of the features. We describe two techniques to improve feature weighting so that more features can be made effective and high quality features can be given more weights.

5.1 Raising Feature Weights

From Eqn. (11), we see that the score of an entity $\mathbf{e}_i$ is determined only by its corresponding feature vector and the weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_J)$. Eqn. (12) shows a value of the weight vector $\mathbf{w}$. We can rewrite Eqn. (12) as follows,

$$w_j = log\frac{\tilde{\alpha}_j}{\alpha_j} - log\frac{\tilde{\beta}_j}{\beta_j}$$

$$= \log\left(1 + \frac{\sum_{i=1}^{N} q_{ij}}{km_j}\right) - \log\,(1 + \frac{N-\sum_{i=1}^{N} q_{ij}}{k(1-m_j)}) \qquad (13)$$

In Eqn. (13), N is the number of items in the seed set. As mentioned before, m_j is the mean of feature j of all possible entities and k is a scaling factor (which we use 1). m_j can be regarded as the prior information empirically set from the data.

In order to make a positive contribution to the final score of entity $\mathbf{e}$, w_j must be greater than zero. Under this circumstance, we can obtain the following inequality based on Eqn. (13).

$$\sum_{i=1}^{N} q_{ij} > Nm_j \qquad (14)$$

Eqn. (14) shows that if feature j is effective ($w_j > 0$), the seed data mean must be greater than the candidate data mean on feature j. Only such kind of features can be regarded as high-quality features in Bayesian Sets. Unfortunately, it is not always the case due to the idiosyncrasy of the data. There are many high-quality features, whose seed data mean may be even less than the candidate data mean. For example, in our drug data set, "prescribe" can be a left first verb for an entity. It is a very good entity feature. "Prescribe EN/NNP" (EN represents an entity, NNP is its POS tag) strongly suggests that EN is a drug. However, the problem is that the mean of this feature in the seed set is 0.024 which is less than its candidate set mean 0.025. So if we stick with Eqn. (14), the feature will have negative contribution, which means that it is worse than no feature at all. The fact that all pattern features are from sentences containing seeds, a candidate entity associated with a feature should be better than no feature.

We propose to tackle this problem by fully utilizing all features found by template patterns. We change original m_j to $\tilde{m}_j$ by multiplying a scaling factor t to force all feature weights $w_j > 0$:

$$\tilde{m}_j = t\, m_j \qquad (0 < t < 1) \qquad (15)$$

The idea is that we lower the candidate data mean intentionally so that all the features found from the seed data can be utilized. In other words, we let $\frac{\sum_{i=1}^{N} q_{ij}}{m_j}$ to be greater than N for all features j. Since N is a constant, we only need to find the minimum value $\frac{\sum_{i=1}^{N} q_{ik}}{m_k}$ of all features, and lower m_k to $\tilde{m}_k$ to make $\frac{\sum_{i=1}^{N} q_{ijk}}{\tilde{\tilde{m}}_k} > N$. Under such condition, the parameter t for Eqn.15 is determined by $\frac{\tilde{m}_k}{m_k}$. Note that although this may also make some bad features have $w_j > 0$, good features' weights will increase even more.

5.2 Identifying High-Quality Features

Eqn. (13) shows that besides m_j value, w_j value is also affected by the sum $\sum_{i=1}^{N} q_{ij}$. It means that if the feature occurs more times in the seed data, its corresponding w_j will also be high. However, Eqn. (13) may not be sufficient since it only considers the feature frequency (as features are binary) but does not take feature quality into consideration. For example, we have two different features A and B, which have the same feature frequency in the seed data and thus the same mean, According to Eqn. (13), they should have the same feature weight w. However, for feature A, all feature count may come from only one entity in the seed set, but for feature B, the feature counts are from four different entities in the seed set. Obviously, feature B is a better feature than feature A simply because the feature is shared by or associated with more entities.

To detect such high-quality features to increase their weights, we use the following formula to change the original w_j to w_j' .

$$w_j' = rw_j \qquad (16)$$

$$r = (1 + \frac{\log h}{T}) \qquad (17)$$

In Eqn. (16), r is used to represent feature quality for feature j. h is the number of unique entities that have jth feature. T is the total number of entities in the seed set.

6. ENTITY RANKING

Although Bayesian Sets produces a rank of test vectors, due to the way that we generate the data (each entity produces multiple feature vectors depending on the number of times that it appears in the corpus), the ranking produced by Bayesian Sets is not a ranking of entities. Since different vectors representing the same entity can have very different scores, this leads to the problem of how to rank the entities, which is the final result that we need. Thus, we need to compute a score for each unique entity. Before we discuss further, let us describe another issue, the entity frequency. As mentioned in the introduction section, it is highly desirable to rank those correct and frequent entities at the top because they are more important than the infrequent ones in opinion mining (or even other applications). With this in mind, we define a new ranking algorithm.

Let the score values of a candidate entity **e** be $V = \{v_1, v_2 \ldots, v_n\}$ obtained from the feature vectors representing the entity. Let M_d be the median value of V and M_m be the mean value of V. We use two steps to design the formula.

Step1: Compute an entity score without considering frequency.

Let the score of a candidate entity **e** be $S(\mathbf{e})$. We use the median as its score, i.e., $S(\mathbf{e}) = M_d$. This is justified based on the statistical criteria *skewness*. The following inequalities hold in general,

$$\begin{cases} M_m \geq M_d & if\ skewness \geq 0 \\ M_m < M_d & if\ skewness < 0 \end{cases} \tag{18}$$

where $skewness = \frac{m_3}{\sqrt{m_2}^3}$

$$m_3 = \frac{\sum(v_i - \bar{v})^3}{n},\ m_2 = \frac{\sum(v_i - \bar{v})^2}{n} \text{ and } \bar{v} = \frac{\sum v_i}{n}.$$

If $skewness < 0$, we should use the median M_d. The intuition is that if the values in V are skewed towards the high side (negative skew), it means that the candidate entity is very likely to be a true entity, and we should take the median as it is also high. However, if the skew is towards the low side (positive skew), it means that the candidate entity is unlikely to be a true entity and we should again use the median as it is low under this condition.

Step2: Final score function considering the frequency

Factoring in the frequency, we have the final score function $S'(\mathbf{e})$ for candidate entity **e**,

$$S'(\mathbf{e}) = S(\mathbf{e})log(1 + f(\mathbf{e})) \tag{19}$$

where $f(\mathbf{e})$ is the frequency count of entity **e**. 1 is added to smooth the value. The idea is to push the frequent candidate entities up by multiplying the log of frequency. Log is taken in order to reduce the effect of big frequency count numbers.

7. THE OVERALL ALGORITHM

Finally, we put everything together to present the final algorithm that we use. We can still improve the algorithm in two ways.

1. Enlarging the seed set using some high precision syntactic coordination patterns.
2. Iteratively running the Bayesian Sets algorithm with bootstrapping. This gives the final algorithm.

We discuss them below.

7.1 Enlarging the Seed Set

One obvious way to improve the results is to increase the number of seeds. However, to do this automatically we need a very high precision method so as not to introduce non-entities into the seed list. Extraction based on syntactic *coordination* patterns is such a method. Coordination patterns are linguistic structures (e.g., "*and*", "*or*") which connect two or more entities together in a sentence. It is a strong indicator that entities are of the same type or class [8][10][13]. However, it also subjects to some noise. For example, the linguistic structure such as "… *X*, *Y*, …" may occur often without implying the same class of *X* and *Y* [29], but introducing apposition or clarification instead. In order to maximally prevent noisy data from being included into the expanded seed set, we need to utilize this technique with care. We only use the following 5 coordination patterns. Again, EN refers to an entity name. '|' is the logical OR relation. '*' stands for zero or more. Additionally, we also require that EN be a proper noun or a phrase of all proper nouns.

P1 EN [or | and] EN
P2 from EN to EN
P3 neither EN nor EN
P4 prefer EN to EN
P5 [such as | especially | including] EN (, EN)* [or | and] EN

Some example sentences are, "*Nokia and Samsung do not produce smart phones*," "*Neither Nokia nor Samsung produce big screen phones*" and "*LCD Brands such as Sony, Philips and Samsung, …*"

The extraction works by matching these patterns. If anyone of the ENs in the patterns is a seed in the seed list, the other matched ENs are extracted. The process can be run iteratively as the newly discovered entities can be used to match the patterns again and find more entities. The process stops after no more new entities are found. This process usually stops in no more than 3 iterations.

```
Algorithm: IterativeBayesianSets(Q, D, T)
  Input: A small seed set Q of user-provided entities;
         A data corpus T containing a collection of documents;
         A set of candidate entities D extracted from T using the
            method in Section 4.1.
  Output: A ranked list of entities in D – Q

1   Q+ ← Expand Q using T and the method in Section 7.1;
2.  D' ← Prepare data (feature vectors) using the method in
         Section 4.3, i.e., each appearance of a candidate entity
         in D – Q+ in a sentence in T forms a feature vector;
3   Q'+ ← Prepare data (feature vectors) using the method in
         Section 4.3, i.e., each appearance of an entity in Q+ in a
         sentence in T forms a feature vector;
4   R ← BayesianSets(Q'+, D');              // in Fig. 2
5   E ← Rank entities in D – Q based on the scores in R using the
         method in Sections 5 and 6, i.e., all candidate entities in
         D are ranked except those in the original Q.
6   A ← Pick up the top k entities in E;     // we use k = 5
7.  Q* ← Q+ ∪ A;
         // Bootstrapping Bayesian Sets below
8.  while (Q* ≠ Q+) do     // New seeds are added
9.      Q+ ← Expand Q* using T and the method in Section 7.1;
10.     Q'+ ← Collect all feature vectors for the entities in Q+;
11.     R ← BayesianSets(Q'+, D');            // // in Fig. 2
12.     E ← Rank entities in D – Q based on the scores in R using
            the method in Sections 5 and 6, i.e., all candidate
            entities in D are ranked except those in the original Q.
13.     A ← Pick up the top k entities in E;  // we use k = 5
14.     Q* ← Q+ ∪ A
15. end while
16. Output the final ranked list E
```

Fig. 3. The iterative Bayesian Sets learning algorithm

7.2 Bootstrapping Bayesian Sets

This strategy again tries to find more seeds, but using Bayesian Sets itself. Thus, we run the Bayesian Sets iteratively. At the end of each iteration, we pick up *k* top ranked entities (we use *k* = 5 in our experiments). Note that the original seeds provided by the user are not in the rank. This newly extract *k* entity set is unioned with the current seed set. The iteration ends if no new entity is added to the current seed list, e.g., the union is the same as the current seed set. The detailed algorithm is given in Fig. 3. The algorithm is self-explanatory. We will not discuss it further.

Table 1. Experimental Data sets or Corpora

Data Set (corpus)	Cars	Insurance	Drug	Stove	Mattress	Cellphone	Blu-ray	Vacuum	Online Banking	LCD
# of sentences	2223	12456	1504	25060	13233	15168	7108	13521	17441	1783

Table 2. Precision of top 15 (3 seeds)

	Car Honda, A4,Toyota	**Insurance** Cobra, Cigna, Kaiser	**Drug** Humira, Enbrel, methotrexate	**Stove** Kenmore, Frigidaire, GE	**Mattress** Simmons, Serta, Heavenly	**Cellphone** Motorola, Nokia, N95	**Blu-ray** Sony, Samsung, s300	**Vacuum** dc17, hoover, roomba	**Online Banking** Citi, Chase, Wesabe	**LCD** Samsung, PZ77U, Sony	**Avg.**
Distr.S-fq	0.800	0.800	0.733	0.866	0.733	0.400	0.266	0.866	0.466	0.200	0.613
S-EM	0.933	0.866	0.666	0.933	0.666	0.733	0.666	0.533	0.533	0.733	0.726
BaS-1-vec	0.800	0.800	0.600	0.866	0.666	0.400	0.266	0.866	0.466	0.466	0.620
BaS-no-fw	0.800	0.733	0.666	0.833	0.733	0.600	0.533	0.933	0.600	0.800	0.723
BaS-no-se	0.866	0.733	0.733	0.866	0.766	0.533	0.600	0.933	0.666	0.733	0,743
BaS-all	0.866	0.800	0.800	0.866	0.800	0.666	0.600	0.933	0.666	0.800	0.779

Table 3. Average precision@15, 30, 45 (3 seeds)

	Distr.S	Distr.S-fq	S-EM	BaS-1-vec	BaS-no-fw	BaS-no-se	BaS-all
Precision @ 15	0.553	0.613	0.726	0.620	0.723	0.743	0.779
Precision @ 30	0.536	0.543	0.650	0.540	0.620	0.643	0.686
Precision @ 45	0.495	0.493	0.617	0.510	0.568	0.597	0.626

8. EXPERIMENTS

This section evaluates the proposed technique. We first describe the data sets, evaluation metrics and then the experimental results. We also compare it with a popular distribution similarity based method and a PU learning method S-EM, and show the effects of individual methods proposed in the paper. Finally, we compare our results with those of two Web based set expansion systems, Google Sets and Boo!Wa!.

8.1 Data Sets

We used 10 diverse data sets to evaluate our technique. They were obtained from a commercial company that provided opinion mining services to its industrial clients. The data were crawled and extracted from multiple online message boards and blogs discussing different types of products and services. Table 1 shows the domains (based on their names) and the number of sentences in each data set. We split each message board post into sentences and the sentences are POS-tagged using the Brill's tagger [5]. The tagged sentences are the input to our system.

8.2 Evaluation Metrics

Since we do not have gold standard entity sets to compare with, regular evaluation metrics for NER such as precision and recall are not suitable for our purpose. We adopt the following metric for evaluation. Precision @ N: The percentage of correct entities that are among the top N entities in a ranked list. This is a commonly used method for set expansion [22].

8.3 Experimental Results

We now present the experimental results in Tables 2 and 3, and Fig. 4. Table 2 shows the precisions of top 15 results (i.e., precisions @15) based on three random seeds (we study the effect of different numbers of seeds below). In the table, the results of seven different methods are listed and compared. The last column shows the average of each row. The average results of top 15, 30 and 45 are shown in Table 3. Fig. 4 presents the same average results as in Table 3 for easy understanding. Below, we first describe each system and then discuss the results.

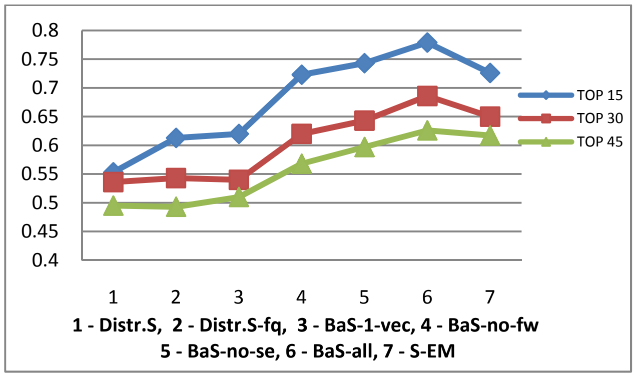

Fig.4. Average precisions of different methods

Dist.S-fq: It denotes the *distributional similarity* method with the entity frequency considered as in Bayesian Sets. Without considering the frequency, it gives poorer results (see **Dist.S** in column 1 of Table 3). We implemented the latest model in [22], which represents each entity as a *Pointwise Mutual Information* vector, and the similarity is compared using the *cosine* measure. For a fair comparison, we also added our word features (Section 4.2), which give better results.

S-EM: It denotes a PU learning method S-EM, which is applied in [18]. We adopt the same word features in [18], i.e. we use 3 words right before the seed or the candidate and 3 words right after it as word features.

BaS-1-vec: It denotes the method that each entity is represented as a single feature vector (the direct application of Bayesian Sets as described in Section 1).

BaS-no-fw: It denotes the proposed method **BaS-all** (see below) only without using the two feature reweighting techniques discussed in Section 5.

BaS-no-se: It denotes the proposed method **BaS-all** only without seed expansion using coordination patterns and bootstrapping (Section 7).

BaS-all: It denotes the proposed technique that utilizes all the new methods.

From Table 2, Table 3 and Fig. 4, we can draw the following conclusions.

1. The distributional similarity methods (**Dist.S** and **Dist.S-fq**) perform poorly. Bayesian Sets with all our enhancements (**BaS-all**) is better by 14-16%. Direct application of Bayesian Sets with one feature vector per entity **(BaS-1-vec)** is also weak. The reasons have been given in Section 1.
2. **S-EM** outperforms **Dist.S-fq** by about 11%. But **BaS-all** outperforms **S-EM**, especially for top 15 results.
3. Comparing **BaS-no-fw** and **BaS-1-vec**, we can see that the new way of applying Bayesian Sets (**BaS-no-fw**: each entity with multiple feature vectors) produces much better results.
4. From the results of **BaS-no-se** (feature reweighting is used but no seed expansion) and **BaS-no-fw**, we can see that the two feature reweighting methods are very helpful.
5. Comparing BaS-no-se and BaS-all (Table 3), we observe that expanding seeds and bootstrapping Bayesian Sets (Section 7) improve the results too.

In summary, we can conclude that the proposed technique (**BaS-all**) is highly effective. The more improvements that we apply to Bayesian Sets, the better result we can achieve. Our full method **BaS-all** gives the best results.

8.4 The Effect of Seed Size

Fig. 5 gives the results of 2, 3, 4, 5 seeds. Clearly, with more seeds, we can achieve better results. However, from the figure, we also observe that the improvements are minor. The main reason is that we have already applied the seed set enlargement step in the algorithm using coordination patterns (Section 7.1) and using bootstrapped Bayesian Sets (Section 7.2).

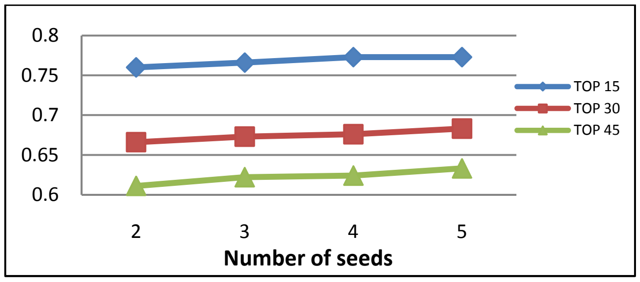

Fig.5. Effects of number of seeds on precision

Table 4. Average precision comparison of our system BaS-all, Google Sets and Boo!Wa! (3 seeds)

	Top 15	**Top 30**	**Top 45**
BaS-all	0.779	0.686	0.626
Google Sets	0.692	0.609	0.560
Boo!Wa!	0.686	0.599	0.511

8.5 Compare with Google Sets and Boo!Wa!

Strictly speaking Google Sets (http://labs.google.com/sets) and Boo!Wa! (http://boowa.com/) are not comparable with our work. They are both Web based systems and use the "whole" Web to expand the seeds set. More importantly, we are only interested in entities discussed in a specific opinion corpus, not everything on the Web. For example, for cars, Google Sets returns major brands of cars, but does not find many specific models, which are of limited use for our purpose. Also, Google Sets mainly find those official names of entities but in our corpora people use all types of short forms. For example, Motorola may be written as Moto and Samsung may be written as Sammy. Product model names have even more variations. Without extracting them, it will result in major losses in opinion mining. Despite these, our method still performs better than Google Sets and Boo!Wa! (Table 4). On average over the 10 seed sets (each has 3 seeds), our system outperforms both Google Sets and Boo!Wa! by a large margin.

9. CONCLUSIONS

This paper studied the entity set expansion problem in the context of opinion documents. For opinion mining, without knowing entities that opinions are expressed on the opinions are of limited use. The classic method of solving the problem is based on distributional similarity. This approach is effective in finding some very frequent entities but is weak for finding less frequent entities. The reason is that infrequent entities have less contextual information. In our work, we employed the Bayesian Sets approach. However, directly applying it gives poor results. We thus proposed a more sophisticated way to use it, which, however, caused two major problems: entity ranking and feature sparseness. We have proposed effective methods to solve the problems. Additionally, we also proposed a set of generic features used for learning, which have been shown effective for a diverse set of domains. This is crucial for scaling-up opinion mining applications as it is too time-consuming to design features for each individual application. Extensive experiments based on 10 commercial data sets (forum discussions and blogs) show that the proposed method outperforms a recent distributional similarity method, the direct application of Bayesian Sets, a PU learning method S-EM, Google Sets and Boo!Wa! by large margins.

10. REFERENCES

[1]. Agichtein E. and Gravano S. Snowball: Extracting relations from large plain-text collections. ACM ICDL, 2000.

[2]. Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., and Soroa, A. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. NAACL HLT, 09.

[3]. Banko, M. Cafarella, M. Soderland, S. Broadhead, M. Etzioni, O. 2007. Open information extraction from the web. IJCAI, 2007.

[4]. Bhole, A., Fortuna, B., Grobelink, M. and Mladenic.,D, Extracting named entities and relating them over time based on wikipedia. *Informatica*. 2007

[5]. Brill, E. Transformation-Based error-Driven learning and natural language processing: a case study in part of speech tagging. *Computational Linguistics*, 1995.

[6]. Bunescu, R. and Mooney, R. Collective information extraction with relational markov networks. ACL, 2004.

[7]. Cheng T., Yan X. and Chang C. K. EntityRank: searching entities directly and holistically. VLDB, 2007.

[8]. Cimiano, P., Handschuh, S., and Staab, S. Towards the self-annotating web. WWW, 2004.

[9]. Downey, D., Broadhead, M. and Etzioni, O. Locating Complex Named Entities in Web Text. IJCAI, 2007.

[10]. Etzioni, O., Cafarella, M., Downey. D., Popescu, A., Shaked, T., Soderland, S., Weld, D. Yates, A. Unsupervised

named-entity extraction from the Web: An Experimental Study. Artificial Intelligence, 165(1):91-134, 2005

[11]. Ghahramani, Z and Heller, K.A. Bayesian sets. NIPS 2005.

[12]. Google Sets. System and methods for automatically creating lists. US Patent: US7350187, March 25, 2008.

[13]. Hearst, M. Automatic acquisition of hyponyms from large text corpora. COLING, 1992.

[14]. Heller, K. and Ghahramani, Z. A simple bayesian framework for content-based image retrieval. CVPR, 2006.

[15]. Jiang, J. and Zhai, C. Exploiting domain structure for named entity recognition. HLT-NAACL, 2006.

[16]. Lafferty J., McCallum A., and Pereira F. Conditional random fields: probabilistic models for segmenting and labeling sequence data. ICML, 2001.

[17]. Lee, L. Measures of distributional similarity. ACL, 1999.

[18]. Li, X., Zhang, L., Liu, B., and Ng, S. Distributional Similarity vs. PU learning for Entity Set Expansion. ACL, 2010

[19]. Lin, D. Automatic retrieval and clustering of similar words. COLING/ACL, 1998.

[20]. Liu, B. Sentiment analysis and subjectivity. Handbook of Natural Language Processing, Second Edition, (editors: N. Indurkhya and F. J. Damerau), 2010.

[21]. McCallum, A. and Li, W. Early results for named entity recognition with conditional random fields, feature induction and enhanced lexicons. CoNLL, 2003.

[22]. Pantel, P., Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, Vishnu, Vyas. Web-Scale Distributional similarity and entity set expansion, EMNLP, 2009.

[23]. Pantel, P. and Lin, D. Discovering word senses from text. KDD, 2002.

[24]. Pang, B. and Lee, L. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval 2(1-2), pp. 1–135, 2008.

[25]. Paşca, M. Weakly-supervised discovery of named entities using web search queries. CIKM, 2007.

[26]. Paşca, M. Lin, D. Bigham, J. Lifchits, A. Jain, A. Names and similarities on the web: fast extraction in the fast lane. ACL, 2006.

[27]. Riloff, E., Jones, R. Learning dictionaries for information extraction by multi-level bootstrapping. AAAI, 1999.

[28]. Sarawagi, S. "Information extraction," Foundations and Trends in Databases, 1(3), pp. 261-377, 2008.

[29]. Sarmento, L., Jijkuon, V. de Rijke, M. and Oliveira, E. 2007. "More like these": growing entity classes from seeds. CIKM, 2007.

[30]. Wang, R.C. and Cohen, W. W. Iterative set expansion of named entities using the web. ICDM, 2008.

[31]. Xu, G., Yang, S. and Li, H. Named entity mining from click-through data using weakly supervised latent dirichlet allocation. KDD, 2009.

[32]. Zhu, J., Nie, Z., Liu, X., Zhang, B. and Wen, J. StatSnowball: a statistical approach to extracting entity relations. WWW, 2009.