Entity Discovery and Assignment for Opinion Mining Applications

Xiaowen Ding
Department of Computer Science
University of Illinois at Chicago
851 S. Morgan Street
Chicago, IL 60607-0753
xding@cs.uic.edu

Bing Liu
Department of Computer Science
University of Illinois at Chicago
851 S. Morgan Street
Chicago, IL 60607-0753
liub@cs.uic.edu

Lei Zhang
Department of Computer Science
University of Illinois at Chicago
851 S. Morgan Street
Chicago, IL 60607-0753
lzhang3@cs.uic.edu

ABSTRACT

Opinion mining became an important topic of study in recent years due to its wide range of applications. There are also many companies offering opinion mining services. One problem that has not been studied so far is the assignment of entities that have been talked about in each sentence. Let us use forum discussions about products as an example to make the problem concrete. In a typical discussion post, the author may give opinions on multiple products and also compare them. The issue is how to detect what products have been talked about in each sentence. If the sentence contains the product names, they need to be identified. We call this problem *entity discovery*. If the product names are not explicitly mentioned in the sentence but are implied due to the use of pronouns and language conventions, we need to infer the products. We call this problem *entity assignment*. These problems are important because without knowing what products each sentence talks about the opinion mined from the sentence is of little use. In this paper, we study these problems and propose two effective methods to solve the problems. Entity discovery is based on pattern discovery and entity assignment is based on mining of comparative sentences. Experimental results using a large number of forum posts demonstrate the effectiveness of the technique. Our system has also been successfully tested in a commercial setting.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Information filtering*. I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *text analysis*.

General Terms

Algorithms, Experimentation.

Keywords

Entity Discovery, Sentiment Analysis

1. INTRODUCTION

It is now well recognized that the user-generated content (e.g., product reviews, forum discussions and blogs) contains valuable consumer opinions that can be exploited for many applications. There are already many companies that provide opinion mining services. In this paper, we report a system that solves an important problem in these applications. It identifies what entities (e.g., products) each sentence talks about.

Most opinion mining researches are based on product reviews [2, 4, 10, 18, 19, 22] because a review usually focuses on a specific product or entity and contains little irrelevant information. However, in forum discussions and blogs, the situation is very different, where the authors often talk about multiple entities (e.g., products), and compare them. This raises two important issues: (1) how to discover the entities that are talked about in a sentence and (2) how to assign entities to each sentence because in many sentences entity names are not explicitly mentioned, but are implied. We term the first problem *entity discovery* and the second problem *entity assignment*. Without knowing the entities that a sentence talks about, any opinion mined from the sentence is of little use. For example, if an algorithm finds that a sentence expresses a negative opinion about something, but it cannot determine on what product, then the opinion is meaningless.

The first problem is basically the named entity recognition (NER) problem. However, traditional NER methods do not work well because of the ungrammatical nature of the forum posts, over-capitalization and under capitalization. Over-capitalization means that the user may capitalize every word in the sentence, and under-capitalization means that the first letters of many entity names are not capitalized. These cause serious problems for existing entity recognition programs as we will see in Section 5. We propose a technique to solve the problem based on sequential pattern mining and natural language processing (NLP).

The second problem is similar to pronoun resolution [3, 24] in NLP, which identifies what each pronoun in a sentence refers to. Pronoun resolution is still a major challenge. The accuracy of the current state-of-the-art systems is only about 60-70% on well-formed sentences such as those in news articles [3, 24]. However, this accuracy cannot be used for applications. In addition, for the user-generated content, the problem is harder due to ungrammatical sentences, and missing or wrong punctuations.

To solve this problem, our proposed method will not rely on pronoun resolution because our task is also different. Many sentences do not have pronouns, but we still need to know which entities these sentences talk about. For example, sentences (3) and (5) of the discussion post in Example 1 below have no pronoun or any other reference to resolve. The question is how to discover the entity that sentences (3) and (5) talk about in Example 1.

Example 1: "(1) I bought Camera-A yesterday. (2) I took some pictures in the evening in my living room. (3) The images are very clear. (4) They are definitely better than those from my old Camera-B. (5) The battery is very good too."

A simple approach to assigning entities that are talked about in

KDD'09, June 28– July 1, 2009, Paris, France.

each sentence is the following: The algorithm sequentially processes each sentence. Whenever an entity name is encountered in a sentence, it is assumed that the sentence talks about that entity. It is also assumed that the subsequent sentences talk about that entity as well until a new entity name appears. Then the new entity is the one talked about in its sentence. The subsequent sentences also talk about the new entity, and so on. This simple strategy works reasonably well in practice. However, it breaks down when a comparative sentence is encountered.

Let us use this strategy to process the sentences in Example 1. Clearly, sentences (1) – (3) talk about Camera-A because Camera-A is encountered in sentence (1). Sentences (2) and (3) contain no new product. Sentence (4) is a comparative sentence, which also introduced Camera-B. Now, the above strategy is not applicable because although sentence (4) only mentions Camera-B, it actually talks about both cameras. What is more serious is that the simple strategy also infers that sentence (5) talks about Camera-B, which is clearly wrong. For us human beings, we can see that sentence (5) talks about Camera-A. The question is how to solve the problem using an automated approach. One may say that the sentence after the comparative sentence should talk about the product mentioned before. Unfortunately, this is not right either. For example, we change Example 1 to:

Example 2: "(1) I bought Camera-A yesterday. (2) I took a few pictures in the evening in my living room. (3) The images were very clear. (4) They were definitely better than those from my old Camera-B. (5) The pictures of that camera were blurring for night shots, but for day shots it was ok"

Example 2 is the same as example 1 except the last sentence. Obviously, sentence (5) here talks about Camera-B. Then, the above simple algorithm does not work.

An important finding of this work is that the implicit entity assignment problem is mainly caused by comparative sentences as Example 1 shows. Based on our data sets, if we simply infer that the implicit entity is the previously mentioned entity, 20% of them are wrong and 98% of these errors are caused by comparisons.

This paper proposes a technique to solve the problem. The method does not rely on pronouns, which has two advantages. First, there is no need to solve the difficult problem of pronoun resolution in NLP. Second, sentences that do not use pronouns can be handled.

Sentiment consistency: To introduce the proposed technique, let us make an important observation about the two examples above. The comparative sentences (4) in both examples say that Camera-A is superior to Camera-B. The next sentence, sentence (5) in Example 1, expresses a positive sentiment. Intuitively, sentence (5) in Example 1 should refer to the superior product. Similarly, sentence (5) in Example 2, which expresses a negative sentiment in its first clause, should refer to the inferior product. We call this phenomenon *sentiment consistency*, which says that consecutive sentiment expressions should be consistent with each other. It would be ambiguous if this consistency is not observed in writing.

It turns out that this observation is quite general, which suggests a novel approach to solving the problem. That is, opinion mining can be employed. Two tasks are necessary: (1) for a comparative sentence, we need to identify which entity is superior, and (2) for the subsequent sentence, we need to determine whether its first clause (sentence 5 of Example 2) is positive, negative, or neutral. The opinion mining technique in [4] is adapted to solve the problem. In the process, we also made two contributions to opinion mining. First, we discovered that the opinion mining method can also be adapted to analyze comparative sentences. Thus, a separate algorithm is not needed although the two types of sentences look quite different (e.g., sentences (4) and (5) in the examples). Second, we define a rule specification language to make the rule-based opinion mining technique in [4] more convenient. Experimental results based on 753 forums posts from 63 threads with 4385 sentences show that our technique is effective. Our system has also been successfully tested in a commercial setting.

2. RELATED WORK

Related works about entity discovery is mainly in the field of named entity recognition (NER). NER aims to identify entities such as names of persons, organizations and locations in natural language text. Most NER methods are based on supervised learning [21]. Recent strategies also exploit the domain structure in the training data to improve the performance [10]. [11] studied extraction of entities and other items from comparative sentences based on comparative sentence structures. On product extraction, [6] used hand-crafting rules to extract product entities and exploited comparative sentences for product comparison analysis. Our work is more general and not focused on comparative sentences. Our work is also different from the classic NER as we are only interested in the product type of entities. [21] gave good surveys of existing NER and general information extraction algorithms. Conditional random fields (CRF) [15] have been shown to perform the best so far. We will show that the proposed method outperforms CRF dramatically for our task.

Although the problem of assigning entities referred to in sentences is important in practice for applications, we are not aware of any focused study of this problem. Related works are mainly in the area of pronoun resolution or co-reference resolution in NLP.

Pronoun or coreference resolution has been extensively studied in natural language processing [3, 17, 24]. However, it is still a major challenge. As discussed in the introduction, our task is in fact quite different because many sentences do not have pronouns, but we still need to know which entities they discuss.

Although the objective of the proposed problem is not opinion mining, some of the opinion mining techniques are applied to solve the problem. The most widely studied opinion mining topic is sentiment classification of documents and sentences, i.e., classifying them as expressing positive, negative, or neutral opinions [2, 18, 22]. However, our work needs to study clauses. In the same sentence, one clause may be positive but another clause may be negative, e.g., "*The photo quality is great, but the battery sucks*", which contains two opinions.

We have made use of some methods in feature-based opinion mining [4, 16, 19] in our work. Feature-based opinion mining means to find opinions expressed on individual product features. For example, in the above example, "photo quality" and "battery" are product features. The opinion on "photo quality" is positive and the opinion on "battery" is negative. Existing techniques exploit opinion words for the task. Opinion words are words that express desired or undesired states. Positive words express desired states, e.g., "great" and "good". Negative words express undesired states, e.g., "bad" and "poor". Identifying opinion words have been studied in [5, 9, 12, 13, 14]. Several lists have been compiled. Apart from individual words, there are also many

opinion phrases, e.g., "cost someone an arm and a leg".

We adapted the method in [4] for our purpose as it can be easily changed for opinion analysis of each clause. We also made two contributions to opinion mining. First, [4] hard-codes all opinion phrases in the system, which is undesirable because any addition/deletion of phrases will involve changing the program code. A specification language is then proposed to allow the user to add/delete phrases without touching the underlying program. Second, we show that our adapted opinion mining method can be easily extended to deal with comparative sentences, which is important for our task of entity assignment.

On the study of comparative sentences, [1, 11] proposed some methods to study comparative and superlative sentences. However, they do not determine superior entities expressed in comparative sentences, which is what we need. [7] studied opinions in comparative sentences. However, it needs a large volume of external information, i.e., pros and cons in product reviews. Our method is much simpler, and we only use the method to solve the entity assignment problem, which is the focus of this paper.

3. THE PROBLEM

The basic information unit of forums, blogs and discussion boards consists of a *start post* and a list of *follow-up* posts or *replies*. This basic information unit is often called a *thread*. A thread t thus can be modeled as a sequence of posts, $<p_1, p_2, \ldots, p_n>$. p_1 is the start post. Each post consists of a sequence of sentences, $<s_1, s_2, \ldots, s_m>$. Each sentence s_i describes something on a subset of entities $\varepsilon = \{e_i \ldots e_j | e_i, e_j \in E\}$, where E is the set of all entities. An *entity* can be a person, a product, an organization, an event, etc. If an entity name is explicitly mentioned in sentence s_i, we say that the entity is an *explicit entity* in s_i. If the entity is not explicitly mentioned in s_i but it is implied, we say that the entity is an *implicit entity*. For example, Camera-A in the first sentence below is an explicit entity. Camera-A is an implicit entity in sentence 2 as it is not explicitly mentioned there, but it is implied.

"*Camera-A looks really pretty. The battery lasts very long*".

Most sentences talk about a single entity, i.e., the size of ε is usually 1. If a sentence involves multiple entities (explicit and/or implicit), it is usually a comparative sentence, e.g., "*Camera-A is better than Camera-B*". A related type of sentences is the superlative sentences, e.g., "*Camera-A is the best*."

This is a simplified model of the real word. For example, it does not cover irrelevant sentences, which are usually rare. It does not cover quotes (from previous posts) in the reply. However, quotes are easy to handle because they are usually in a different format.

Problem statement: Given a set of threads T in a particular domain, two tasks are performed in this paper:

1. *Entity discovery*: discover the set of entities E discussed in the posts of the threads, and
2. *Entity assignment*: assign the entities in E that each sentence s_i of each post p_j in t ($\in T$) talks about.

4. ENTITY DISCOVERY

The main idea for entity discovery is to discover linguistic patterns and then use the patterns to extract entity names. However, traditional methods need a large number of manually labeled training examples, and labeling is very time consuming. For a different domain, the labeling process may need to be repeated. This section proposes an automated pattern discovery method for the task, which is thus unsupervised.

The basic idea of the algorithm is that the user starts with a few seed entities. The system bootstraps from them to find more entities in a set of documents (or posts). The algorithm is thus iterative. Sequential pattern mining [8] is employed at each iteration to find more entities based on already found entities. The iterative process ends when no new entity names are found. Pruning methods are also proposed to remove those unlikely entities. Given a set of seed entities $E = \{e_1, e_2, \ldots e_n\}$, the algorithm consists of the following iterative steps:

Step 1 – data preparation for sequential pattern mining: This step perform two tasks, it first finds all sentences that contain anyone of the seed entities, e_1, e_2, …, e_n in the dataset, and then generate a sequence for each occurrence of e_i for pattern mining. In order to focus patterns on entities and not generate too many patterns, we use only a window of 5 words before each entity name and 5 words after each entity name. Each word of a seed entity name is replaced with a generic name "ENTITYXYZ". The purpose of using this generic word is to ensure that general patterns about any entities are found. Note that each entity name may consist of more than one word. The part-of-speech (POS) tag of each word is also used. In the final sequence each element of the sequence is a pair, POS tag of the word and the word.

Example 3: We have the following sentence with POS tags attached. Here n95 is a phone model (an entity).

Hiiiiiiii/NNP SK/NNP -/: ,/, dont/NN be/VB mad/JJ everyone/NN doesnt/NN have/VBP a/DT n95/CD phone/NN fetish/NN ducky/JJ

The window is (n95 has been replaced with ENTITYXYZ):

mad/JJ everyone/NN doesnt/NN have/VBP a/DT ENTITYXYZ /CD phone/NN fetish/NN ducky/JJ

The resulting sequence is:

<{JJ, mad}{NN, everyone}{NN, doesnt}{VBP, have}{DT, a}{CD, ENTITYXYZ}{NN, phone}{NN, fetish} {JJ, ducky}>

Step 2 – Sequential pattern mining: Given the set of sequences generated from step 1, a sequential pattern mining algorithm is applied to generate sequential patterns [8]. We use 0.01 as the minimum support. We also require that each pattern must contain {POStag, ENTITYXYZ} and its length to be greater than or equal to 2 for obvious reasons. An example pattern is:

<{IN}, {DT}, {NNP, ENTITYXYZ }, {is}>

Here "IN", "DT", "NNP" are POS tags which can match any words with that tag, and "is" is a concrete word which can only match this particular word.

Step 3 – Pattern matching to extract candidate entities: For each sentence in the test dataset, the system matches the generated patterns to extract a set of candidate entities. The patterns are sorted based on their supports. In order not to generate too many spurious candidates, the matching process in a sentence terminates after 5 patterns have been matched. We tried several numbers and find that 5 is a good number with respect to results and efficiency.

Example 4: We have the follow sentence with POS tags attached:

The/DT misses/VBZ has/VBZ currently/RB got/VBN a/DT Nokia/NNP 7390/CD at/IN the/DT end/NN of/IN the/DT day,/VBG all/DT she/PRP does/VBZ is/VBZ text/NN and/CC

make/VB calls,/NN but/CC the/DT reception/NN is/VBZ terrible,/VBG where/WRB my/PRP$ 6233/CD would/MD get/VB full/JJ bars/NNS hers/PRP would/MD only/RB get/VB 1/CD or/CC 2./CD

The pattern, <{DT}, {NNP, ENTITYXYZ}, {CD}>, will match the sentence segment, a/DT Nokia/NNP 7390/CD, to produce the candidate entity: "Nokia". The pattern, <{DT}, {NNP}, {CD, ENTITYXYZ}, {IN}>, will match the sentence segment, a/DT Nokia/NNP 7390/CD at/IN, to produce the candidate entity: 7390

Step 4 – Candidate pruning: The above pattern matching may extract many wrong entities. A pruning method based on POS check is proposed. It remedies some errors made by the POS tagger system. Since an entity is always associated with a POS tag in our patterns, this method checks in the dataset to see whether the POS tag is the most frequent one for this candidate. If it is not, the candidate entity is eliminated (a possible POS tagging error).

Example 5: Given the sentence:

You/PRP can/MD also/RB be/VB sure/JJ it/PRP will/MD work/VB with/IN all/PDT the/DT Sony/NNP Ericsson/NNP walkman/NN phone/NN accessories/CD

The pattern, <{IN}{DT}{CD, ENTITYXYZ}>, matches the sentence segment:

with/IN all/PDT the/DT Sony/NNP Ericsson/NNP walkman/NN phone/NN accessories/CD

to produce the candidate entity: accessories, which is incorrect.

But when the algorithm goes over the sentences in the dataset again, it found that "accessories" appear as "NNS" more often than as "CD". This candidate is deleted.

The algorithm so far is generic and applicable to any domain because no assumption was made. The step below is more applicable to manufactured products (which are our main area of applications), which have brands and models. It should not be used for non-manufactured products. This step makes the assumption that a model name has a digit in it. In the experimental section we will show their results separately.

Step 5 – Pruning using brand and model relation and syntactic patterns. For most manufactured products, brands and models often appear together, e.g., "Moto Razr V3". Here we need to use the above digit assumption. Thus, based on the entities that were found so far (step 4), this step tries to prune entities by using the pattern <Brand Model>. The first task is to discover relationships from the entities discovered so far. This is simple as the example below shows.

Example 6: We have the following sentence:

As/RB far/RB as/IN I/PRP heard/VBD Nokia/NNP N95/CD seems/VBZ to/TO be/VB the/DT leader/NN in/IN this/DT sense./CD

In this sentence, if both "Nokia" and "N95" are in the entity list, "Nokia" is considered as <Brand>, and "N95" is considered a <Model>.

Then using some syntactic patterns can help find competing brands and models. The syntactic patterns exploit conjunctions and comparisons in sentences.

We use C to denote a discovered entity and CN as a competitor. The following eight patterns are used:

C and CN	CN and C
C or CN	CN or C
C vs CN	CN vs C
C more than CN	CN more than C

The second task is to remove those entities discover in step 4 that never appear together with a <Band> or a <Model>, or never appear with a candidate in the syntactic patterns.

5. ENTITY ASSIGNMENT

We now present the entity assignment algorithm, which depends on opinion mining of comparative sentences. Below we first introduce the concepts of comparatives and superlatives and then discuss their impacts on the problem. Based on the discussion, the algorithm is naturally derived.

5.1 Comparatives and Superlatives

Comparative and superlative sentences are instrumental to our task. We define them here based on [11].

5.1.1 Comparative Sentences

Comparative sentences express similarity and differences of more than one entity. There are three main types of comparatives,

1) *Non-equal gradable*: "g*reater* or *less than"* that expresses a total ordering of some entities with regard to some shared features or attributes. For example, the sentence, "*Camera-X's battery life is longer than that of Camera-Y*", orders Camera-X and Camera-Y based on their shared feature "battery life".
2) *Equative*: "*equal to"* that states two entities as equal with respect to some features. For example, the sentence, "*Camera-X and Camera-Y are of the same size"*, expresses that the two cameras are equal in term of their shared feature "size".
3) *Non-gradable:* Comparing two or more entities, but do not grade them. For example, the sentence, "*Camera-X and Camera-Y have different shapes*", expresses a comparison of the shapes of the two cameras but does not grade them.

5.1.2 Superlative Sentences

A superlative sentence expresses a relation of the type "*greater* or *less than all others*", i.e., it ranks one entity over all other entities. For example, the sentence, "*Camera-X's battery life is the longest*", expresses a superlative relation.

5.2 Sentiment Consistency

Intuitively, in a post, if the author starts with a particular entity, he/she will continue with the entity. If he/she wants to introduce a new entity *e*, he/she has to state the name of the entity explicitly in a sentence s_0, which can be (1) a normal, (2) a comparative or (3) a superlative sentence. The question is what happens to the next sentence s_1 if s_1 is a normal sentence and does not mention any entity, or s_1 is a comparative sentence and it does not mention *e*.

For (1), when s_0 is a normal sentence, if s_1 is a normal sentence, it should talk about *e*. If s_1 is a comparative sentence, it should compare *e* with a new entity, which should be explicitly mentioned. For (2), when s_0 is a comparative sentence, if s_1 is a normal sentence, there are a few cases:

s_0 is *non-equal gradable*: If s_1 has no entity name and it expresses a positive (respectively negative) sentiment, it should talk about the superior (or inferior) entity to satisfy *sentiment consistency*.

s_0 is *equative*: In this case, it is unclear which entity is referred to in s_1. We assume that it is the previous entity before s_0.

s_0 is *non-gradable*: In this case, it is also unclear which entity is referred to in s_1. It is assumed to be the previous entity talked about before s_0.

For (3), when s_0 is a superlative sentence, if s_1 is a normal sentence, it refers to the *superlative entity* in s_0. For both (2) and (3), if s_1 is a comparative sentence, the entities in s_1 are taken.

5.3 The Algorithm

Based on the above discussion, a natural algorithm emerges, which is given in Figure 1. It follows the simple method given in Section 1 but with special handlings to comparative sentences as discussed above. The input is a post, and the output is a set of entities discussed in each sentence. Note that the algorithm is simplified for presentation clarity. In our implemented system, the start post and quotes in replies are also considered as entities may be inherited from them. Comparative sentences here cover superlative sentences that contain more than one entity. For a superlative sentence with only a single entity, it is treated as a normal sentence. The notations used in the algorithm are:

s_i.Entity: It stores the names of the entities discussed in sentence s_i, which can be explicit or implicit.

s_i.superiorEntity: It stores the set of superior entities in the comparative sentence s_i. Note that we use a set here because the sentence may compare two sets of entities, e.g., "*Camera-A is better than Camera-B and Camera-C*." However, in practice, each set mostly contains only a single entity.

s_i.inferiorEntity: It stores the set of inferior entities in the comparative sentence s_i.

opinion(): It is the opinion mining function that analyzes a non-comparative sentence.

compOpinion(): It is the opinion mining function that finds superior and inferior entities from a comparative sentence.

```
for each sentence s_i in sequence in a post do
1  If s_i is not a comparative sentence then
2     if s_i contains an explicit entity then
3        s_i.Entity ← the explicit entity of the sentence s_i
4     else    // s_i does not contain an explicit entity
5        if s_{i-1} is not a comparative sentence then
6           s_i.Entity ← s_{i-1}.Entity
7        elseif a superior entity and an inferior entity were
                    discovered in s_{i-1} then
8           opinion(s_i);    // opinion mining
9           if s_i's first clause is a positive clause then
10             s_i.Entity ← s_{i-1}.superiorEntity
11          elseif s_i's first clause is a negative clause then
12             s_i.Entity ← s_{i-1}.inferiorEntity
13          else  s_i.Entity ← s_{i-1}.superiorEntity
14       else s_i.Entity ← s_f.Entity, entities of the last sentence
                    that is not a comparative sentence
15 else       // s_i is a comparative sentence
16    if no entity is mentioned in s_i then
17       s_i.Entity ← s_{i-1}.Entity
18    else s_i.Entity ← {s_{i-1}.Entity} ∪ {entities in s_i};
19       ⟨s_i.inferiorEntity, s_i.superiorEntity⟩ ← compOpinion(s_i)
```

Figure 1: The overall algorithm for entity assignment.

6. OPINION MINING

We now present the opinion mining method used in the algorithm (i.e., ***opinion***(s_i)), which is based on the method in [4]. Interestingly, we will also show in Section 6.3 that comparative sentences can be analyzed in a similar way (i.e., ***compOpinion***()).

The main idea of the approach is to use opinion indicators to decide the *orientations* of opinions expressed on entity features. Orientations of opinions mean whether the opinions are positive, negative or neutral. There are three main opinion indicators that are used in opinion mining, i.e., opinion words and phrases, negations, and but-clauses. They are discussed below.

6.1 Opinion Indicators

Opinion words and phrases: In most cases, opinions in sentences are expressed with *opinion words*, e.g., "*great"*, "*good*", "*bad*", and *"poor"*. Researchers have compiled sets of such words. Such lists are collectively called the *opinion lexicon*. In this work, we obtained the list from [4] with some additions. Apart from individual words, there are *opinion phrases* and *idioms*, e.g., "cost someone an arm and a leg". Furthermore, some phrases may involve opinion words, but the whole phrases have no opinion or their opinions depend on contexts. For example, the phrase "a good deal of" does not have an opinion although it has the positive opinion word "great". Such phrases are called *non-opinion phrases involving sentiment words*.

While most adjectives/adverbs have explicit positive or negative orientations, there are also many words whose orientations depend on contexts in which they appear. For example, the word "long" in the following two sentences has completely different orientations, one positive and one negative: "*The battery of this camera lasts long*" and "*This program takes a long time to run.*" A method will be described in Section 6.3 to deal with this.

Negations: opinion words and phrases form the basis of opinions in a sentence. Negations reverse their orientations. Apart from "not", many other words and phrases can be used to express negations. Furthermore, "not" may not express negation in some cases, e.g., in "not only … but also". Such phrases are called *non-negations involving negation words*.

But-clauses: "but" means contrary. For example, the sentence, "*The picture quality is great, but not the battery life*" expresses a positive opinion on "picture quality" but a negative opinion on "battery life". The following rule states the effect of "but":

The orientation before "but" is opposite to that after "but".

Apart from the word "but", many other words and phrases behave similarly, e.g., "though" and "except that". Similar to opinions and negations, not every "but" changes opinion direction. For example, "but" in the pattern "not only … but also" does not. Such phrases are called *non-but phrases involving "but"*.

6.2 Specification for Opinion Indicators

With a large number of indicators, one can hard-code them in a system, which is, however, very undesirable because whenever a new word or phrase is encountered the program needs to be changed, which is time consuming. In [4], all phrases are hard-coded in the system. In this work, we propose a specification language to enable the user to specify indicators, which are (1) *opinion words and phrases*, (2) *negation words and phrases*, (3) *but-like words and phrases*, (4) *non-opinion phrases involving sentiment words*, (5) *non-negation phrases involving negation words*, and (6) *non-but phrases involving but-like words*. The system then automatically uses the indicators for opinion mining (Section 6.3). All the opinion indicators are compiled manually.

Two types of specifications are used: one for individual words and one for phrases. The reason for the separation is that individual

words express their default meanings, but their meanings may be changed by phrases, i.e., overwriting the defaults to express the indicators (4), (5) and (6).

Specification of Individual Words: The grammar of the language for expressing individual words, which include opinion words, negation words and but-like words, is given below:

```
<rule>    :=  <item> "=>" <action>
<item>    :=  <word> | <word> "[" <type> "]"
<word>    :=  [a-z]+
<type>    :=  JJ | RB | NN | VB | …
<action>  :=  Po | Ne | Neu | Ng | But
```

The specification consists of a set of rules, i.e., each indicator word is represented as a rule. Each rule consists of two parts, an *item* on the left and an *action* on the right. The <item> is either an individual word or a word attached with a type, which may be anyone of the part-of-speech (POS) tags. <action> may be anyone of the five symbols, Po (positive), Ne (negative), Neu (neutral), Ng (negation) and But (but-like word). For example, if we want to express that "like" expresses a positive opinion when it is a verb, we can use:

like[VB] => Pos

Given a sentence, the system applies each rule by matching the word together with its type in the sentence and then associates the action symbol to the matched word.

Specification for Phrases: The grammar is given below.

```
<rule>         :=  <pattern> "=>" <action>
<pattern>      :=  <exp> "+" <target> "+" <exp>
                   | <exp> "+" <target> | <target> "+" <exp>
<exp>          := <element> | <exp> "+" <element>
                   | <exp> "+" <distance> "+" <exp>
                   | <exp> "+" <distance>
                   | <distance> "+" <exp>
                   | !<num> "+" !<item> "+" <exp>
                   | <exp> "+" !<num> "+" !<item>
                   |<exp> "+" !<num> "+" !<item> "+" <exp>
<element>      :=  <item> | item "/" element
<item>         :=  <indicator> | <word>
<indicator>    :=  <indicatorSym>
                   | <indicatorSym> "[" <type> "]"
<target>       :=  <indicator> "[T]" | <word> "[T]"
<indicatorSym> := Po | Ne | Neu | Ng | But
<word>         :=  [a-z]+ | [a-z]+ "[" <type> "]"
<distance>     :=  <num> | <num> - <num>
<num>          :=  0 | [1-9][0-9]*
<action>       :=  <outcome>| !<outcome>
<outcome>      :=  PO | NE | NEU | NG | BUT
<type>         :=  JJ | RB | NN | VB | …
```

The specification again consists of a set of rules. Each rule has two parts, a *pattern* on the left and an *action* on the right. Each pattern has a *target word*, indicated by [T], to which the action is applied. The idea is that the left-hand-side of the rule is first matched in the sentence and then the action of the rule is applied to the target in the sentence. See an example in Section 5.3.

Some main concepts used in the grammar are:

IndicatorSym: These are indicator symbols, Po, Ne, Neu, Ng and But, from individual indicator words discussed above. A "type" may also be attached, specifying the POS tag of the word.

Word: It can be any word with an optional type.

Distance: It indicates the number of words (or gap) that can appear between two non-adjacent items in the phrase. "-" means from "num" to "num" (num is an integer number).

Target: It is the core item of the phrase, indicating which word the rule is applied to.

Some additional notes about the grammar: "+" is the separator, "/" means "or" and "!num + !<item>" means that within <num> words gap, <item> does not appear.

The action on the right states that the action symbol should be associated with the target. The action symbol can be any of the outcomes or their negations, i.e., PO (positive), NE (negative), NEU (neutral), NG (negation), and BUT (but-like). "!" means 'not'. These action symbols cannot appear on left-hand-side, which prevents looping.

Some remarks about the language are: The ordering of rules is significant. When the first rule for a target word is matched and applied, the rest will not be tried. Choosing the right target is also important in the situation where a phrase overwrites the default meaning of a word. The target should be the word in question. For example, the rule "great => Po" specifies that "great" is positive. However, the phrase "a great deal of" overwrites the orientation of "great" because "a great deal of" has no opinion. In this case, the rule should be "a great[T] + deal + of => NEU" as the opinion of "great" is nullified by the phrase. If we use "a great deal[T] of => NEU", "great" will still be treated as positive.

6.3 Opinion Mining

We now describe opinion mining. We use the following running example sentence to show the working of each step:

> "*The picture quality of this camera is not good, reaction is too slow, but the battery life is long*."

Step 1 – Part-of-speech tagging: The tags are used for matching <type>'s in the rules.

Step 2 – Applying indicator word rules: All opinion words, negation words and but-like words in the sentence are discovered in this step. For our example, after this step, we obtain

> The picture quality is not**[Ng]** good**[Po]**, reaction is too slow**[Neu]**, but**[But]** the battery life is long**[Neu].**

All the bold attachments are added in this step. The POS tags are omitted to improve readability.

Step 3 - Applying phrase rules: This step identifies all phrases in the sentence and performs the actions specified in the rules. After this step, our running example sentence becomes:

> The picture quality is not[Ng] good[Po], reaction is too slow**[NE]**, but[But] the battery life is long[Neu].

The orientation of "slow" is revised to negative ([NE]) due to the rule: "too + Neu[JJ][T] => NE".

Step 4 - Handling negations: A negation in a sentence reverses the orientation of an opinion. For neutral, it is turned to negative. After negation handling, our running example sentence becomes ("good" is now turned to negative from positive):

> The picture quality is not[Ng] good**[Negative]**, reaction is too slow[NE], but[But] the battery life is long[Neu].

Step 5 - Aggregating opinions: This step first finds but-symbols,

which indicate opinion changes. The opinions on the two sides of a but-symbol are opposite to each other.

Opinion aggregation: All opinion indicators in the first clause of the sentence are aggregated to arrive at the final opinion. The algorithm simply sums up all indicators. A positive (or negative) indicator is assigned 1 (or -1). If the final sum is greater than 0, then the clause is positive. If the sum is less than 0, then the clause is negative and neutral otherwise.

Handling context-dependent opinions: For those sentences that the above process cannot determine their orientations, the algorithm checks if it can detect context dependent opinions as in [9], which uses several rules. Only the conjunction rule is used in this work (the others are inaccurate). For example, in "*The battery life is long*", it is unclear whether "long" means positive or negative. The method tries to see whether any other person said that "long" is positive (or negative). If another person wrote "*this camera takes great pictures and has a long battery life*". From this sentence, we can infer that "long" is *positive* for "battery life" because it is conjoined with the positive word "great". This is the *conjunction rule*, which says that a sentence only expresses one opinion, unless there is a but-like word changing the direction.

6.4 Opinion Mining of Comparisons

As we mentioned earlier, the opinion mining method above can be adapted to find superior and inferior entities in comparative sentences. This is due to the fact that positive and negative opinion words have their corresponding comparative and superlative forms indicating superior and inferior states respectively. For example, the positive opinion word, "good", has its comparative and superlative forms, "better" and "best", which indicate superior (and inferior) entities.

In English, comparatives and superlatives are special forms of adjectives and adverbs. In general, comparatives are formed by adding the suffix "*-er*" and superlatives are formed by adding the suffix "*–est*" to the *base* (or *original*) *adjectives* and *adverbs*. Adjectives and adverbs with two syllables or more and not ending in *y* do not form comparatives or superlatives this way. Instead, "more", "most", "less" and "least" are used before such words, e.g., "more interesting" and "most awful". These two types are called *regular comparatives* and *superlatives*. English also has *irregular comparatives* and *superlatives* that do not follow the above rules. These are, "more", "most", "less", "least", "better", "best", "worse", "worst", "further/farther" and "furthest/farthest".

In order to use the opinion mining method mentioned above to find superior and/or inferior entities, we first convert those opinion adjectives and adverbs to their comparative and superlative forms, which is done automatically by using English grammar rules and WordNet. Due to space limitations, we will not discuss the conversion in detail here as it is fairly straightforward. We then regard the comparatives and superlatives as positive and negative as their base forms respectively. For irregular comparatives, "better" and "best" are treated as positive, and "worse" and "worst" are treated as negative. "more", "most", "less", and "least" require special handling. They are considered together with opinion words using the following 4 rules:

more/most + Pos → Positive
more/most + Neg → Negative
less/least + Pos → Negative
less/least + Neg → Positive

Rule 1 says that "more/most" and a positive (Pos) opinion word together mean positive, e.g., "more beautiful". Other rules have similar meanings.

Non-standard words: Apart from the above comparatives and superlatives, many other words can also express comparisons, e.g., "win", "prefer", "superior" and "inferior". For example, the sentence, "*In term of battery life, Camera-X is superior to Camera-Y*", expresses a comparison indicating that *Camera-X* is preferred with regard to "*battery life*". These words are treated as positive or negative based on their meanings.

Identify comparative and superlative sentences: Before we can identify superior entities from comparative sentences, we need to identify such sentences. [11] proposed a pattern mining approach to identifying comparative and superlative sentences. In this work, we did not focus on this task. Only several heuristic rules are designed to identify such sentences, which perform quite well.

Clearly, comparative and superlative sentences are signaled by various keywords. We use a list of 67 keywords (obtained from [11]), which includes 4 part-of-speech tags, i.e., JJR (comparative adjective), RBR (comparative adverb), JJS (superlative adjective) and RBS (superlative adverb). Our heuristics rules are as follows (if a sentence matches anyone of the rules, it is considered a comparative or a superlative sentence):

a). pronoun + compkey + prodname,
b). prodname + compkey + pronoun,
c). prodname + compkey + prodname
d). pronoun + superkey
e). prodname + superkey
f). as + JJ + as (except "as long as" and "as far as")

where *compkey* is a comparative keyword, *prodname* is a product name and *superkey* is a superlative keyword.

Discover superior entities: Finally, as mentioned earlier, the above opinion mining method can be used to discover superior entities. Since a gradable comparative sentence typically has entities on the two sides of the comparative keyword, i.e., "*Camera-X is better than Camera-Y*". Based on opinion mining, if the sentence is positive, then the entities before the comparative keyword is superior and otherwise they are inferior (with the negation considered). Superlative sentences can be handled in a similar way. Note that equative and non-gradable comparisons do not express preferences.

7. EMPIRICAL EVALUATION

This section evaluates the proposed techniques for the two tasks, entity discovery and entity assignment. Below, we first describe our datasets and then present the experimental results.

7.1 Experimental Data Collections

The experiment data collections are crawled from two forums, HowardForums and AVSforums. HowardForums is a message board dedicated to mobile phones while AVSforum is a message board dedicated to Home Theater and the products used. Our data from AVSforum are discussions about Plasma and LCD TVs, Projectors and DVD players. Table 1 shows the characteristics of the two data sets. Altogether, we downloaded 64 threads, which contain 753 individual posts with 1072 comparative and superlative sentences. The total number of sentences is 4385. All the sentences and product names were annotated by two graduate students based on consensus.

Table 2: Results of entity discovery

Datasets	CRF		NET		EI (1-3)		EI (1-4)		EI (1-5)	
	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
Howard	0.40	0.91	0.48	0.35	0.87	0.48	0.86	0.58	0.81	0.83
	F = 0.56		F = 0.40		**F = 0.62**		**F = 0.69**		**F = 0.82**	
AVS	0.37	0.89	0.42	0.29	0.84	0.47	0.84	0.59	0.77	0.80
	F = 0.52		F = 0.34		**F = 0.60**		**F = 0.69**		**F = 0.78**	

Table 3: Experimental results for entity assignment

	Next Sentences (Accuracy)				All Sentences (Accuracy)				Comp Ident.		
Data sets	Baseline1	Baseline2	ED (k-com)	ED (unk-com)	Baseline1	Baseline2	ED (k-com)	ED (unk-com)	Prec.	Recall	F
HowardForums	82.4%	83.3%	93.4%	90.3%	80.3%	82.1%	88.2%	86.7%	85.2%	84.2%	84.7%
AVSforrum	79.6%	80.9%	91.2%	89.6%	76.7%	77.9%	87.2%	85.0%	82.2%	84.9%	83.5%
Average	81.0%	82.1%	92.3%	89.9%	78.5%	80.0%	87.7%	85.9%	83.7%	84.6%	84.1%
Col#	1	2	3	4	5	6	7	8	9	10	11

7.2 Experimental Results

We now present the experimental results for both tasks.

Table 1: Characteristics of the two data sets (comparative sentences including superlative sentences)

Data sets	**No. of threads**	**No. of posts**	**No. of Product**	**No. of comparatives**	**Total no. of sentences**
Howard	31	446	171	664	2589
AVS	33	307	180	408	1796
Total	64	753	351	1072	4385

7.2.1 Entity Discovery

The results of entity discovery are given first. Our method is called EI. It is compared with the NET system [26] from University of Illinois at Urbana Champion, and the Conditional Random Fields method (CRF) [15]. NET is a Named Entity Tagger, which can be used in our case as product names are named entities. The CRF system that we use is from Sunita Sarawagi [25]. Table 2 shows the results.

Note that the NET system does not need training. The training data for CRF is the data obtained from step 2 of our algorithm. Recall that the data from step 2 is automatically generated. The entities in those sentences are regarded as positive data and all the other words in the sentences are regarded as negative data. The test data is the whole set for all the systems. Using the whole set as the test data is reasonable because our system does not use any manually labeled training data. Only a set of seed entities is supplied. The training data is automatically generated.

In Table 2, we also compare EI when the first 3 steps (EI (1-3)), the first 4 steps (EI (1-4)) and all 5 steps are used (EI (1-5)). Using the first 3 steps basically means that the system only uses pattern mining for extraction. Pruning is step 4 was quite effective. As expected EI(1-4) produces high recalls but low precisions. However, EI(1-4)'s *F* scores are already dramatically higher than those of CRF and NET. For NET, we only used its results for organization entities. For other types of entities, the results are much worse. From Table 2, we also see that the additional step of EI improve the result further (EI(1-5)). Compared to EI(1-4), the precision increases dramatically with a small drop in recall, but the overall *F* scores are much higher. In practice, step 5 (which makes the digit assumption) is not needed because the high recall is the key. As the resulting entity list is not long, the user can filter out those non-entities fairly easily.

Recall that our system uses some seeds to start the process. The question is how the number of seeds affects the final results. We performed a set of experiments by varying the number of seeds to see their effects. Figure 3 gives the results of 5, 10, 15 and 20 seeds. Clearly, when the number of seeds is small the precision is higher, but the recall is very low. With more seeds, we get more balanced results. If more seeds are selected, although the results are slightly better, it defeats the purpose of method which requires little user knowledge. Our experimental results reported in Table 2 are based on 15 seeds. All results are the averages of 10 random runs with randomly selected seeds.

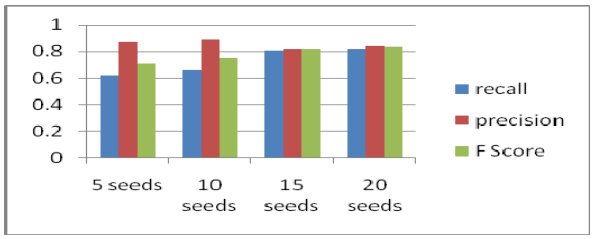

Figure 3: Results of different seeds for entity discovery (Average of the two datasets)

7.2.2 Entity Assignment

Table 3 gives the experimental results for entity assignment, which includes the results of two baseline methods. We use ED to denote the proposed technique. Two sets of experiments were conducted. The first set is denoted by "Next Sentences" in Table 3. "Next Sentences" means that only the comparative sentences and their subsequent sentences are considered. This set of experiments thus shows how effective the ED technique is in its intended task. The second set of experiments is denoted by "All Sentences", which considers all sentences. It shows how the ED method affects the overall implicit entity assignment task.

Column 1 (baseline1-next sentences): Baseline1 works as follows: If a sentence does not mention any product name, we simply take the last product of the previous sentence. Note that the product of the previous sentence may be inherited from its previous sentence and so on. The accuracy measure is used here because we want to gauge how accurate the assignments of products to sentences are.

Column 2 (baseline2-next sentences): In the Baseline2 method, if a sentence does not mention a product name, it simply takes

the first product of the previous sentence.

Discussion: We observe that Baseline2 is always more accurate than Baseline1 because in most cases, the first product is the superior product in a comparative sentence and the next sentence also tends to talk about that product. 98% of the errors are caused by comparative sentences.

Column 3 (ED (k-com) – next sentences): It gives the result of each data set using the proposed ED method assuming that the comparative and superlative sentences are known. *k-com* denotes this assumption.

Column 4 (ED (unk-com) – next sentences): It gives the result of each data set using the proposed ED technique assuming that the comparative and superlative sentences are unknown. *unk-com* denotes this fact. This is the realistic situation, in which the system has to detect comparative and superlative sentences automatically using the method in Section 6.4.

Discussion: We observe that ED outperforms the two baseline methods dramatically, i.e., on average from the best accuracy of the baselines, 82.1%, to the accuracy of the realistic situation of not knowing the comparatives, 89.9%. Knowing the comparative sentences (k-com) only performs slightly better as compared to not knowing them (unk-com). Note that the accuracy here means the total number of sentences that have been correctly assigned products compared to the total number of sentences that need such assignments.

Columns 5-8 (all sentences): These results correspond to those in columns 1-4 except that all sentences are used in the experiments. In this case, the algorithm assigns products to every sentence rather than only to the sentence after each comparative and superlative sentence.

Discussion: Again, we see major improvements, i.e., on average from the best of the baselines, 80.0%, to the realistic situation of not knowing the comparatives, 85.9%. In this case, ED improves slightly less because comparative sentences are only a small proportion of all sentences. The results are lower than columns 1-4 since due to propagation if the discovery in one sentence is wrong, we will get the implicit entity in the next sentence wrong and so on.

Columns 9-11: They give the precision, recall and F-score of each data set on the task of identifying comparative and superlative sentences. The average result (F = 84.1%) is better than that given in [11], i.e., the average F = 79%.

In summary, the experimental results clearly demonstrated the effectiveness of the ED method.

8. CONCLUSION

This paper presented a practical system that deals with two related problems in applications of opinion mining, i.e., mining entities discussed in a set of posts and assigning entities to each sentence. These are so important that without solving them, any opinion discovered from the user-generated content is of limited use. We proposed a pattern-based method to deal with the first problem. To solve the second problem, we first showed that the problem is mainly caused by comparative sentences. We then showed that the problem can be dealt with to a large extent by opinion mining on both the comparative sentences and the subsequent sentences. In the process, we also advanced the state-of-the–art of opinion mining, especially in the analysis of comparative sentences. Our experimental results show that the proposed techniques are effective. In our future work, we will further improve the accuracy of the system.

9. REFERENCES

[1] Bos, J., and Nissim, M. An Empirical Approach to the Interpretation of Superlatives. EMNLP'06*, 2006.*

[2] Dave, D., Lawrence, A., and Pennock, D. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. WWW'03, 2003.

[3] Denis, P., and Baldridge, J, A Ranking Approach to Pronoun Resolution, IJCAI'07, 2007.

[4] Ding, X., Liu, B., and Yu, P. A Holistic Lexicon-Based Approach to Opinion Mining, WSDM'08, 2008.

[5] Esuli, A., and Sebastiani, F. Determining Term Subjectivity and Term Orientation for Opinion Mining, EACL'06, 2006.

[6] Feldman R., Fresko M., Goldenberg J., Netzer O., and Ungar L.: Extracting Product Comparisons from Discussion Boards, ICDM 2007

[7] Ganapathibhotla M., and Liu B. Mining opinions in comparative sentences, Coling-2008.

[8] Han, J., and Kamber M. Data Mining: Concepts and Techniques, 2006

[9] Hatzivassiloglou, V., and McKeown, K. Predicting the Semantic Orientation of Adjectives. ACL-EACL'97, 1997.

[10] Hu. M, and Liu, B. Mining and summarizing customer reviews. KDD-2004.

[11] Jiang, J., and Zhai C. Exploiting Domain Structure for Named Entity Recognition, HLT-NAACL 2006

[12] Jindal, N., and Liu, B. Mining Comparative Sentences and Relations. AAAI'06, 2006.

[13] Kaji, N., and Kitsuregawa, M. Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents. EMNLP'07, 2007.

[14] Kanayama, H., and Nasukawa, T. Fully automatic lexicon expansion for domain-oriented sentiment analysis. EMNLP'06, 2006.

[15] Kim, S., and Hovy, E. Determining the Sentiment of Opinions. COLING'04, 2004.

[16] Lafferty J., McCallum A., and Pereira F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, ICML 2001

[17] Mei, Q., Ling, X., Wondra, W., Su, H., and Zhai, C. Topic Sentiment Mixture: Modeling Facets and Opinions in Weblogs. WWW'07, 2007.

[18] Ng, V. Supervised ranking for pronoun resolution: Some recent improvements. AAAI'05, 2005.

[19] Pang, B., Lee, L., and Vaithyanathan, S. Thumbs up? Sentiment Classification Using Machine Learning Techniques. EMNLP'02, 2002.

[20] Popescu, A.-M., and Etzioni, O. Extracting Product Features and Opinions from Reviews. EMNLP'05, 2005.

[21] Riloff, E., and Wiebe, J. Learning extraction patterns for subjective expressions. EMNLP'03, 2003.

[22] Sarawagi, S. Information Extraction (Survey). Forthcoming.

[23] Turney, P. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. ACL'02, 2002.

[24] Wiebe, J., and Riloff, E. Creating Subjective and Objective sentence classifiers from unannotated texts. CICLing, 2005.

[25] Yang, X, Su, J., and Tan, C.L. Improving Pronoun Resolution Using Statistics-Based Semantic Compatibility Information, ACL'05, 2005.

[26] http://crf.sourceforge.net/

[27] http://l2r.cs.uiuc.edu/~cogcomp/asoftware.php?skey=NE