A Logic Programming Approach to Aspect Extraction in Opinion Mining

Qian Liu*†, Zhiqiang Gao*†, Bing Liu‡ and Yuanlin Zhang§
*School of Computer Science and Engineering, Southeast University, Nanjing, China 210096
†Key Laboratory of Computer Network and Information Integration (Southeast University)
Ministry of Education, Nanjing, China 210096
Email: {qianliu, zqgao}@seu.edu.cn
‡Department of Computer Science, University of Illinois at Chicago
851 South Morgan Street Chicago, IL 60607-7053
Email: liub@cs.uic.edu
§Department of Computer Science, Texas Tech University, Lubbock, Texas 79409
Email: y.zhang@ttu.edu

***Abstract*—Aspect extraction aims to extract fine-grained opinion targets from opinion texts. Recent work has shown that the syntactical approach performs well. In this paper, we show that Logic Programming, particularly Answer Set Programming (ASP), can be used to elegantly and efficiently implement the key components of syntax based aspect extraction. Specifically, the well known double propagation (DP) method is implemented using 8 ASP rules that naturally model all key ideas in the DP method. Our experiment on a widely used data set also shows that the ASP implementation is much faster than a Java-based implementation. Syntactical approach has its limitation too. To further improve the performance of syntactical approach, we identify a set of general words from WordNet that have little chance to be an aspect and prune them when extracting aspects. The concept of general words and their pruning are concisely captured by 10 new ASP rules, and a natural extension of the 8 rules for the original DP method. Experimental results show a major improvement in precision with almost no drop in recall compared with those reported in the existing work on a typical benchmark data set. Logic Programming provides a convenient and effective tool to encode and thus test knowledge needed to improve the aspect extraction methods so that the researchers can focus on the identification and discovery of new knowledge to improve aspect extraction.**

***Keywords*—aspect extraction; logic programming; answer set programming; opinion mining; dependency relation**

I. Introduction

Aspect extraction aims to extract fine-grained opinion targets from opinion texts. An aspect is an attribute or feature of an entity, which can be a product or service. Aspects are important for aspect-based opinion mining (also called sentiment analysis) [1], because without knowing them, the opinions expressed in the texts are of limited use. There are two kinds of aspects [2]: *explicit aspects*, which are explicitly expressed as nouns or noun phases, and *implicit aspects*, which are other types of aspects. For example, the sentence "The price of this car is high" talks about the price of the car. "Price" is an explicit aspect as it is a noun and explicitly appears in the sentence. The sentence "This car is expensive" also talks about the price of the car, but "price" is not in the sentence. "Expensive" here is an implicit aspect, which indicates the explicit aspect "price."

In recent years, explicit aspect extraction has been studied extensively [2]–[11]. There are two main approaches: *statistical* and *syntactical*. The former is mainly based on conditional random fields (CRF) [12], [13] and topic modeling [14]–[24], while the latter is mainly based on dependency relations [25]. Recent work has shown that dependency based methods such as *double propagation* (DP) [26] performs better than CRF and topic modeling. The key idea of the syntactical approach is that opinions have targets (aspects) and there are often explicit syntactic relations between opinion words and aspects. By exploiting some of these relations, the DP method uses a set of seed opinion words to extract aspects and new opinion words and then uses them to extract more aspects and opinion words until no new words can be extracted. Since the process propagates information back and forth between opinion words and aspects, it is called *double propagation*.

However, the discovery of effective syntactic relations and semantic information in the extraction rules usually needs extensive experimentation. In fact, the IBM team of system Watson [27] observed that, to build successful and scalable methods for natural language processing, a huge number of experiments (5500 experiments in 3 years for system Watson) were needed to test the ideas and help conceive new ideas and thus new experiments. Logic Programming (LP) was used in their system to facilitate the experimentation.

We propose to employ logic programming for effective experimentation and building of the key components of aspect extraction systems. Note that basic factual information on the words in the opinion texts and some routine processing are still implemented using traditional language. In this approach, the syntactic relations and the knowledge on how those relations are related to aspect and opinion words are represented naturally as logical rules. The aspects will then be automatically generated by the logic programming inference engines from the logic rules.

To illustrate the logic programming approach, consider a piece of knowledge used in the DP method: "if a word T, whose part-of-speech (POS) is a singular noun (NN), is modified by an opinion word O, then T is an aspect." By this knowledge, in the sentence "The phone has a good screen," "screen" is an aspect because it is a singular noun and modified

by an opinion word "good" (which is known a priori). The knowledge can be represented by the logic rule:

```
aspect(T)  :-  depends(T,mod,O),
               opinionW(O),pos(T,nn).
```

where :- is understood as logic *if*, `aspect(`T`)` denotes that the word T is an aspect, `opinionW(`O`)` that the word O is an opinion word, `pos(`T`,nn)` that the POS of T is NN, `depends(`T`,mod,`O`)` that T is modified by O.

The extraction process inevitably produces errors because it uses only syntactical information. In many existing works [21], [26], infrequent candidate aspects are pruned. However, this method may result in significant loss in precision or recall. Typically, a threshold is used to tell whether a frequency is high or low. As a result, to improve the precision, we need to raise the threshold, which will hurt the recall, and vice versa. To improve precision and recall, methods other than simple frequency threshold have to be used.

We observed that there is a large class of words which are so general that in very few cases they are aspects. Normally, we will not take these words as aspects. As an example, in "I can't write enough good things about this camera," "things" is extracted as an aspect because it is modified by the opinion word "good." However, "things" is very unlikely to be a product aspect and thus should be pruned. We propose to use WordNet [28] to automatically generate a list of general words using three typical general words "thing," "person," and "place" as seeds. By extending the DP method with the knowledge that a general word is normally not an aspect, we obtain a major improvement in the precision with almost no drop in recall on a widely used benchmark data set.

In summary, we make two contributions: (1) We propose to employ Answer Set Programming (ASP) – a variant of Logic Programming – to implement syntactical approach based aspect extraction. Our implementation of the DP method is more elegant and efficient, and it has only 8 rules, while a Java implementation has about 510 lines of code. The ASP based implementation can process about 3000 sentences per second, while the Java implementation only processes about 300 sentences per second. The preciseness and simplicity of the logic programming rules enable the sharing of knowledge used in aspect extraction and the reproducibility of experimental results. (2) We introduce the concept of general words based on WordNet and augment the DP method with the knowledge that general words normally should not be taken as aspects, which results in more accurate aspect extraction. Again, the general words and new knowledge can be naturally implemented using ASP.

The remaining of the paper is organized as follows: we present background and related work in Section II and an overview of our logic programming approach in Section III. The ASP rules to implement the DP method for extracting explicit aspects are described in Section IV. Our new approach to aspect pruning is presented in Section V. We present the experiments in Section VI and conclude the paper and discuss future work in Section VII.

II. Background and Related Work

In this section we introduce the basics of aspect extraction and Answer Set Programming.

A. Aspect Extraction

An *object* is an entity which can be a product, service, person, event, organization, or topic. It is associated with a set of components or attributes, called *aspects* of the object. Each component may have its own set of aspects.

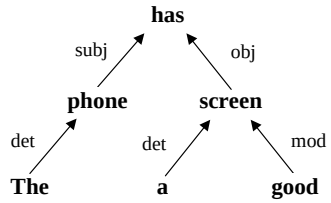
For example, a particular brand of cellular phone, say iPhone, is an object. It has a set of components, e.g., battery and screen, and also a set of attributes, e.g., voice quality, size, and weight. These components and attributes are aspects of the phone.

An *opinion* is simply a positive or negative view, attitude, or emotion about an object or an aspect of the object from a person or an organization. Given a collection of opinion texts on an object, the *aspect extraction* problem is to produce the aspects of the object from these documents.

As mentioned earlier, there are two main methods for aspect extraction. In this paper, we focus only on the syntactical approach as it has been shown to perform better than the statistical approach [26]. For related work on the statistical approach, please refer to the recent book [1]. In the syntactical approach, explicit aspect extraction consists of two phases: *candidate aspect extraction* and *incorrect aspect pruning*.

For *candidate aspect extraction*, we focus on the double propagation method [26] which is based on the following observations. The first is that it is easy to identify (a priori) a set of opinion words such as "good" and "bad," etc. The next is that opinion words are usually associated with aspects (opinion targets) under certain syntactic relations. For example, in the sentence "This camera is good," "good" is an opinion word. The "camera," a noun modified by "good," is clearly an aspect. Therefore from a given set of opinion words, we can derive a set of aspects in terms of syntactic relations. Similarly, syntactic clues can help extract new aspects from the extracted aspects, and new opinion words from the extracted aspects. This propagation process continues until no more opinion words or aspects can be extracted.

Dependency grammar is adopted to represent the syntactic relations used in the propagation. See the picture below for an example of the dependency tree for the sentence "The phone has a good screen."



A *direct dependency* indicates that one word depends on another word without any additional words in their dependency path or they both depend on a third word directly. The DP method considers only direct dependencies as complex relations can make the method vulnerable to parsing errors. Opinion words are assumed to be adjectives and aspects nouns or noun phrases. Thus the potential POS tags for opinion words are JJ (adjectives), JJR (comparative adjectives) and JJS (superlative adjectives) while those for aspects are NN (singular nouns) and NNS (plural nouns). The dependency

relations between opinion words and aspects include *mod*, *pnmod*, *subj*, *s*, *obj*, *obj2* and *desc* whose meaning is shown in Table I. The relations between opinion words and between aspect words contain only the conjunction relation *conj*. The relation *det* means determiner.

TABLE I. DESCRIPTION OF DEPENDENCY RELATIONS.

Relation	Description
mod	the relationship between a word and its adjunct modifier
pnmod	post nominal modifier
subj	subject of verbs
s	surface subject
obj	object of verbs
obj2	second object of ditransitive verbs
desc	description

DP method employs eight rules to derive aspects from opinion words and vice versa in terms of their dependency relations. Details of these rules are given in Section IV.

In the *incorrect aspect pruning* phase, infrequent candidate aspects are pruned in many existing works [2], [26], [29]. Alternative methods, such as pointwise mutual information score [3], pattern-based filters [26] and named entity recognition [30] are also employed to improve the accuracy.

B. Answer Set Programming

Answer Set Programming originates from non-monotonic logic and logic programming. It is a logic programming paradigm based on the answer set semantics [31], [32], which offers an elegant declarative semantics to the negation as failure operator in Prolog. An ASP program consists of *rules* of the form: l_0 :- $l_1, ..., l_m, not\ l_{m+1}, ..., not\ l_n$. where each l_i for $i \in [0..n]$ is a literal of some signature, i.e., expressions of the form $p(t)$ or $\neg p(t)$ where p is a predicate and t is a term, and *not* is called *negation as failure* or *default negation*. A rule without body is called a *fact*.

The rule is read as: if one believes $l_1, ...,$ and l_m and there is no reason to believe $l_{m+1}, ...,$ or l_n, one must believe l_0. The *answer set semantics* of a program P assigns to P a collection of answer sets, i.e., interpretations of the signature of P corresponding to possible sets of beliefs (i.e., literals). These beliefs can be built by a rational reasoner by following the principles that the rules of P must be satisfied and that one shall not believe anything unless one is forced to believe.

A literal is *ground* if it does not contain any variables. A *ground* ASP program is a program containing only ground literals. The *answer set* of a ground ASP program without negation as failure is the minimal model of the program. The *reduct* of a ground ASP program Π with respect to a set S of ground literals, denoted by Π^S, is a program obtained from Π by removing all rules containing *not* l such that $l \in S$ and removing from the rest of the rules *not* and the literal following it. Note that the reduct of a program contains no negation as failure operators. A set S of ground literals is an *answer set* of a ground ASP program Π if S is the answer set of Π^S.

Any rule r with variables of a program Π can be taken as a set of its ground instantiations, i.e., it can be seen as the set of rules obtained from r by replacing its variables by ground terms of the signature of Π. Therefore, an ASP program with variables is a shorthand of a ground ASP program.

A typical reasoning task for ASP program is to find its answer sets while a task for a Prolog (classical LP) program is to answer a query. Most modern inference systems for the former, often called *ASP solvers*, are based on DPLL algorithm [33] – a backtracking based search algorithm – equipped with state-of-the-art reasoning algorithms for propositional logic such as watched literals based unit propagation and learning clauses [34]. The inference system for the latter is usually based on SLD resolution.

ASP can be used to represent and reason with recursive definitions, defaults with strong and weak exceptions [32], causal relations, beliefs, intentions, incomplete information, and some other constructs of natural language. ASP has a solid mathematical theory as well as a number of efficient ASP solvers including the popular solver CLASP [35] and commercial solver DLV [36]. These solvers and the ASP programming methodology have been employed to provide competitive solutions to numerous problems from planning, natural language processing, space shuttle control, product configuration to bioinformatics and other areas [37].

III. A LOGIC FRAMEWORK FOR ASPECT EXTRACTION

An ASP based framework of aspect extraction consists of the following steps:

1) Develop algorithms (usually in a conventional language such as C) to extract the primitive relations in terms of syntactic, semantic and other analysis, and represent them as ASP facts;
2) Identify opinion words and opinion expressions and represent them as ASP facts;
3) Identify extraction rules and other common sense knowledge about the primitive relations and opinion expressions, and represent them by ASP rules;
4) Compute the answer set of the logic program resulted from the first three steps using existing ASP solvers. The aspects are then extracted from the answer set.

In addition to our familiarity with ASP, we choose ASP with the following reasons. (1) The rules used in the syntactical approach can be naturally represented by ASP rules. Furthermore, there are exceptions to these rules. The non-monotonicity of ASP provides an excellent way to represent and reason with exceptions. (2) The existing ASP solvers such as DLV [36] and CLASP [35] offer very efficient reasoning for ASP programs. They are very well documented and maintained, and used by a good number of researchers and/or practitioners[1]. (3) The fast unit propagation mechanism [34] employed by modern ASP solvers can efficiently handle the ASP rules we need to represent the DP method and other knowledge in aspect extraction.

The main objective of this paper is to examine the effectiveness of ASP in aspect extraction over the conventional approach. There are numerous variants of Logic Programming languages and systems (e.g., Prolog and defeasible logic [38]).

[1]For DLV, see http://www.dlvsystem.com/, and for CLASP, see http://potassco.sourceforge.net/

Some of them may be equally good as or better than ASP. A further investigation of other Logic Programming variants is beyond the scope of this paper.

IV. Explicit Aspect Extraction Using ASP

In this section, we discuss in detail how to use ASP to represent the knowledge used in the DP method.

In the DP method, a list of seed opinion words are given a priori. They are represented as facts in the form `opinionW(`T`)` denoting that T is an opinion word. The information about the POS of words and dependency relations between pairs of words in the corpus are automatically extracted by a parser such as Stanford Parser[2]. They are represented as facts `pos(`T, P`)`, which denotes that the POS of T is P, and `depends(`H, Dep, T`)`, which denotes that T depends on the word H through the dependency relation Dep. The other knowledge used in DP is the eight extraction rules that establish relations between opinion words and aspects as well as among opinion words and among aspects themselves through dependency relations.

The first rule in DP [26] is $R1_1$: if $O \rightarrow O\text{-}Dep \rightarrow T$, such that $O \in \{O\}$, $O\text{-}Dep \in \{MR\}$, where $\{MR\}$ = {*mod*, *pnmod*, *subj*, *s*, *obj*, *obj2*, *desc*} and $POS(T) \in \{NN\}$, then T is an aspect. It means "if a word T, whose POS is NN, is directly depended by an opinion word O through one of the dependency relations *mod*, *pnmod*, *subj*, *s*, *obj*, *obj2* and *desc*, then T is an aspect." This rule can be represented as an ASP rule:

```
aspect(T)  :-  depends(T,O-Dep,O),
               opinionW(O),pos(T,nn),
               O-Dep!=conj.
```

where `aspect(T)` denotes that the word T is an aspect, `depends(T,O-Dep,O)` that O depends on T through $O\text{-}Dep$ (except *conj*), `opinionW(O)` that O is an opinion word, and `pos(T,nn)` that the POS of T is NN. Since the dependency relation in this extraction rule covers all dependency relations except the relation *conj*, we have `O-Dep!=conj` in the ASP rule.

In the following, we will intuitively explain each rule in English instead of giving the original rules in [26]. Rule $R1_2$ in DP is: "if an opinion word O and a word T, whose POS is NN, directly depend on a third word H through dependency relations except *conj*, then T is an aspect." This rule can be represented as follows:

```
aspect(T)  :-  depends(H,O-Dep,O),
               depends(H,T-Dep,T),
               opinionW(O),pos(T,nn),
               O-Dep!=conj,T-Dep!=conj.
```

Rule $R2_1$ in DP says "if a word O, whose POS is JJ (adjective), directly depends on an aspect T through dependency relations except *conj*, then O is an opinion word." It is represented as follows:

```
opinionW(O)  :-  depends(T,O-Dep,O),
                 aspect(T),pos(O,jj),
                 O-Dep!=conj.
```

[2]http://www-nlp.stanford.edu/software/lex-parser.shtml

Rule $R2_2$ in DP, which means "if a word O, whose POS is JJ, and an aspect T, directly depend on a third word H through relations except *conj*, then O is an opinion word" can be represented as follows:

```
opinionW(O)  :-  depends(H,O-Dep,O),
                 depends(H,T-Dep,T),
                 aspect(T),pos(O,jj),
                 O-Dep!=conj,T-Dep!=conj.
```

Rule $R3_1$ in DP, which means "if a word T_j, whose POS is NN, directly depends on an aspect T_i through *conj*, then T_j is an aspect" can be represented as follows:

```
aspect(Tj)  :-  depends(Ti,conj,Tj),
                aspect(Ti),pos(Tj,nn).
```

Rule $R3_2$ in DP, which means "if a word T_j, whose POS is NN, and an aspect T_i, directly depend on a third word H through the same dependency relation, then T_j is an aspect" can be represented as follows:

```
aspect(Tj)  :-  depends(H,Ti-Dep,Ti),
                depends(H,Tj-Dep,Tj),
                aspect(Ti),pos(Tj,nn),
                Ti-Dep==Tj-Dep.
```

Rule $R4_1$ in DP, which means "if a word O_j, whose POS is JJ, directly depends on an opinion word O_i through *conj*, then O_j is an opinion word" can be represented as follows:

```
opinionW(Oj)  :-  depends(Oi,conj,Oj),
                  opinionW(Oi),pos(Oj,jj).
```

Rule $R4_2$ in DP, which means "if a word O_j, whose POS is JJ, and an opinion word O_i, directly depend on a third word H through the same dependance relation, then O_j is an opinion word" can be represented as follows:

```
opinionW(Oj)  :-  depends(H,Oi-Dep,Oi),
                  depends(H,Oj-Dep,Oj),
                  opinionW(Oi),pos(Oj,jj),
                  Oi-Dep==Oj-Dep.
```

As we can see, all the 8 extraction rules in DP can be represented naturally as ASP rules. Here is an example to illustrate how our logic framework works for aspect extraction.

Example 1: From the sentences "The phone has a good screen" and "I can't write enough good things about this camera," we obtain, using a parser, the following facts about syntactical information, denoted by program P_1:

```
f1   pos(phone,nn).
f2   pos(good,jj).
f3   pos(screen,nn).
f4   pos(camera,nn).
f5   pos(things,nn).
f6   depends(screen,mod,good).
f7   depends(has,subj,phone).
f8   depends(has,obj,screen).
f9   depends(write,subj,i).
f10  depends(things,mod,good).
f11  depends(write,obj,things).
```

Next, a number of seed opinion words have been identified (e.g., as in [2]). Let P_2 be the program of all rules of the form `opinionW(`W`)` where W is a seed opinion word. In the example sentences, "good" is a seed opinion word. P_2 contains

f_{12} `opinionW(good).`

Let P_3 be the program of the eight ASP rules whose underlying knowledge is identified manually.

According to rule $R1_1$ and facts f_3, f_6 and f_{12}, "screen" is an aspect, i.e., `aspect(screen)` is believed. According to the same rule and facts f_5, f_{10} and f_{12}, `aspect(things)` is also believed.

Finally, we run an ASP solver to compute answer sets of $P = P_1 \cup P_2 \cup P_3$. The solver will output a unique answer set which consists of all atoms occurring in P_1 and P_2 (e.g., `opinionW(good)` from P_2 and `depends(screen,mod,good)` from P_1), and atoms `aspect(camera)` and `aspect(things)`[3].

It is not difficult to see that the ASP extraction rules are a formal specification of the relations between aspects and opinion words that are at the core of DP method. Given F, the facts about seed opinion words and facts about the syntactical information (e.g., program P_1) of the words in a collection of opinion texts, a word W is recognized as an aspect by DP method if and only if `aspect(`W`)` is in the answer set of the program Π that consists of the ASP extraction rules and the facts of F. A brief explanation on why the claim holds is as follows. All the ASP extraction rules do not have negation as failure operator. It is a well known result in Logic Programming that a program without *not* has a unique answer set. If W is an aspect by DP method, there must be a sequence of words $W_1, \cdots, W_n$, where W_n is W, such that W_1 is a seed opinion word, and W_i ($i \in [2..n]$) is an aspect (or opinion) word with the reason that W_{i-1} is an opinion (or aspect) word according to the extraction rules and syntactical information on W_{i-1} and W_i that is in F. Therefore, the words $W_1, \cdots, W_n$ and syntactical information among them satisfy all the ASP extraction rules and facts in Π. Furthermore, since W_1 is an opinion word and thus a fact of F, we must have W_i ($i \in [1..n]$) as an aspect or opinion word in the answer set, i.e., the minimal model, of Π according to the well known properties of answer set programs without negation as failure. Hence `aspect(`W`)` is in the answer set of Π. Similarly, we can prove the other direction of the claim on the aspect W.

To find the answer set of an ASP program is an NP-complete problem. The worst case complexity of ASP solvers is exponential. However, given that the ASP extraction rules and facts on syntactical information do not have negation as failure operators, the watched literals based unit propagation of ASP solvers is sufficient to find an answer set of those rules and facts. Watched literals based unit propagation has a linear complexity. Hence a modern ASP solver can find the answer set of the ASP program for DP method in linear time.

It is also worth noting the difference between a traditional approach – DP method – to aspect abstraction and the declarative approach by ASP. DP method has to describe both the propagation algorithm and the extraction rules while ASP program contains only the rules. The propagation mechanism of DP method is implied by the answer set semantics of the ASP program for aspect extraction.

[3]As one can imagine, the answer set can be very big. In fact, special language constructs in e.g., CLASP allow us to hide all other literals that were used during the computation.

V. Incorrect Aspect Pruning

The extracted aspects inevitably have errors, i.e., incorrect aspects. For example, in "I can't write enough good things about this camera," by rule $R1_1$, "things" is an aspect because it is modified by the opinion word "good." However, "things" is very unlikely to be a product aspect and should be pruned. The precision of the DP method is expected to be improved if it is augmented with incorrect aspect pruning. In this section, we present a new method to identify incorrect aspects and ASP rules for the augmented DP method.

A. Incorrect Aspect Identification

Many incorrect aspects are domain independent, and they are usually general words such as "situation," "someone" and "day," etc. The question we try to answer is how to produce a reasonable set of general words. Explicit aspects are nouns or noun phrases. The definition of "noun" in WordNet 2.0 is "a word that can be used to refer to a *person* or *place* or *thing*." Our intuition is to take *person, place,* and *thing* as the seeds for general words and expand them automatically. This intuition is formalized by the following inductive definition.

Words "thing," "person" and "place" are *general words*; given a general word T, if the first sense of T in WordNet has synonyms, every noun synonym of T is a *general word*, otherwise every noun in the definition of T's first sense is a *general word*. A word may have many senses; we use the first one to avoid over propagation of general words. In the rest of the paper, the definition of a word refers to the definition of its first sense in WordNet for simplicity.

Normally, a word is an incorrect aspect if it is or its definition contains a general word. For example, the word "user," whose definition is "a *person* who makes use of a thing," is an incorrect aspect as the definition includes a general word "person." However, there are two types of exceptions. The first type includes the words whose definition contains "aspect," "attribute," "property," or "component." They are very likely to be product aspects. For example, the word "quality," whose definition is "an essential and distinguishing *attribute* of something or someone," is an aspect despite the definition contains the general word "someone." The second type includes the candidate aspects with high frequency in the collection of opinion texts of concern, obtainable from the DP method, which is described in Section IV, and the words whose definition contains one of the frequent candidate aspects.

Given a word T, if T has noun synonyms in its first sense in WordNet, we obtain the facts `syn(`W,T`)` denoting W is a synonym of T in T's first sense in WordNet, otherwise we obtain the facts `noun(`W,T`)` denoting that W is a noun in T's definition. We can also obtain the facts `highFrequent(`T`)` denoting T is a candidate aspect with high frequency obtainable from a collection of opinion texts by the DP method.

Now we are in a position to write the ASP program for our knowledge on incorrect aspects (denoted by predicate `non_aspect`).

```
non_aspect(T) :- generalWord(T),
                 not exception(T).
```

```
non_aspect(T)  :-  noun(W,T),
                   generalWord(W),
                   not exception(T).
exception(T)   :-  highFrequent(T).
exception(T)   :-  noun(W,T),
                   highFrequent(W).
exception(T)   :-  noun(W,T),
                   isIndicator(W).
```

where `not` is used to exclude the exceptions (`exception`), and `isIndicator(W)` holds if W is a word "aspect," "attribute," "property," or "component."

Our definition of *general word* can be naturally represented by the following ASP rules.

```
generalWord(thing).
generalWord(place).
generalWord(person).

generalWord(T)  :-  syn(W,T),
                    generalWord(W).
generalWord(T)  :-  noun(W,T),
                    generalWord(W).
```

Some examples of general words are shown in Table II.

TABLE II. EXAMPLES OF DOMAIN INDEPENDENT GENERAL WORDS EXTRACTED FROM WORDNET 2.0.

thing	person	place	date	sun
state	people	region	day	event
situation	someone	world	earth	brain
occasion	importance	woman	time	mind
instance	significance	location	month	calendar

As in Example 1, "things" in "I can't write enough good things about this camera" is extracted as a candidate aspect. Since "things" is a general word with no exceptions, we can get the fact `non_aspect(things)` according to the above logic rules.

B. DP Augmented with Non-aspect Pruning

There are three steps in our ASP-based aspect extraction. First, run the logic program of the DP method to produce candidate aspects from a given collection of opinion texts. A conventional program is then used to identify high frequent candidate aspects from the texts. Second, run the logic program for non-aspect identification in this subsection together with the facts on high frequent candidate aspects, to get non-aspect facts. Third, run the augmented logic program with non-aspect pruning to obtain the final aspects.

After identifying non-aspect facts, we then need to revise the ASP extraction rules to take into account the impact of non-aspects using default negation. For example, the ASP rule for $R1_1$ is modified as:

```
aspect(T)  :-  depends(T,O-Dep,O),
               opinionW(O),pos(T,nn),
               O-Dep!=conj,
               not non_aspect(T).
```

which reads "a word T with POS of NN is believed to be an aspect if T is depended by an opinion word O through O-Dep (except *conj*) and there is no reason to believe T is an incorrect aspect."

All other ASP extraction rules that define predicate `aspect/1` should be modified by adding, to their bodies, `not non_aspect(T)` or `not non_aspect(T`$_j$`)` depending on the variable (T or T_j) used in the head of the rules.

When we look back at Example 1, we now have the new fact `non_aspect(things)` (intuitively meaning that "things" is not extracted as a candidate aspect), i.e., "things" is pruned naturally, using the above augmented logic rules.

Note that our approach of modifying the ASP extraction rules is different from simply removing the non-aspect words from the aspects resulted from original ASP extraction rules, due to the propagation effect. For example, in the sentence "I like to take pictures of beautiful people and buildings," "people" is pruned as it is a known non-aspect, "buildings" is also pruned although it is not a known non-aspect word thanks to the propagation effect.

VI. EXPERIMENTS

Experiments are designed to examine the effectiveness of the proposed logic programming approach to aspect extraction in terms of the effort needed to develop aspect extraction systems and their efficiency, and the evaluation of the new aspect pruning method.

We use the customer review collection built by [2] in our experiments, which contains review data sets for five products: two digital cameras (D1, D2), one cell phone (D3), one MP3 player (D4), and one DVD player (D5). Explicit aspects in these review data sets are already labeled. The detailed information about each data set is shown in Table III. For the seed opinion words, we use those provided by [2]. The average number of facts, including POS facts, dependency relation facts and opinion word facts, for a data set is 11,985. The logic program for each dataset has a unique answer set.

TABLE III. DETAILED INFORMATION OF THE DATA SETS.

Data	Product	#of Sentences	#of Labeled aspects
D1	Canon	597	237
D2	Nikon	346	174
D3	Nokia	546	302
D4	Creative	1716	674
D5	Apex	740	296
Avg		789	337

A. Development Effort Comparison

The main difference between the traditional implementation [26] and the logic programming implementation of the DP method lies in the implementation of extraction rules. To implement DP, our java code has about 510 lines. In contrast, only 8 rules are used in our ASP implementation of the DP method.

To identify incorrect aspects, we only use 10 additional rules. To augment the DP method with non-aspect pruning, we simply modify the original 8 rules. Clearly, ASP programs are much smaller and yet every ASP rule is simple and has a natural interpretation, which significantly shortens the development time and improves the maintainability of the system. Another immediate benefit is that it is much easier to reproduce the logic-based system than the traditional one.

B. Efficiency of Aspect Extraction Systems

We use the ASP solver CLASP 3.0.4 to run our ASP program of the DP method on a Windows 7 desktop computer with 3.30 GHz Intel (R) Core (TM) i3-2120 CPU and 4 GB RAM to test the efficiency. Our program is able to process 3,000 sentences per second. On the same computer, our Java implementation of the DP method extracts the same candidate aspects, but it is 10 times slower. The efficiency of our ASP implementation can be explained as follows. Given the nature of the DP method, the watched literals based unit propagation [34], [35] underlying CLASP offers fast implementation (linear complexity) of our ASP program. In contrast, our Java implementation is not equipped with the advanced propagation methods (e.g., watched literals) due to their complexity.

C. Quality of DP with New Aspect Pruning Method

We compare the DP method with infrequent candidate aspect pruning method used in [26], denoted by DP, with the DP method augmented with new aspect pruning method presented in this paper, denoted by DPAP. Table IV shows the precision (the positive predictive value), recall (fraction of retrieved relevant aspects), and F-score of DP and DPAP. DPAP has the highest F-score. We can observe that DPAP has much higher precision than DP thanks to the proposed pruning method, i.e., candidate aspects that are general words can be pruned in DPAP while cannot be guaranteed in DP. The recall of DPAP is slightly lower than DP because some correct aspects are also pruned (for details, see the later discussion in this subsection). On average over the 5 data sets, the precision of DPAP is 16% higher than that of DP. The average recall of DPAP only drops by 2% compared with DP. These show that the proposed pruning is effective. In terms of F-score, DPAP consistently outperforms DP on every data set. On average, DPAP's F-score is 8% higher than that of DP.

TABLE IV. PRECISION, RECALL AND F-SCORE OF DP AND DPAP FOR EXPLICIT ASPECT EXTRACTION.

Data	Precision		Recall		F-score	
	DP	DPAP	DP	DPAP	DP	DPAP
D1	0.70	0.90	0.90	0.83	0.79	0.86
D2	0.74	0.87	0.89	0.86	0.80	0.87
D3	0.77	0.90	0.90	0.90	0.83	0.90
D4	0.69	0.84	0.89	0.86	0.78	0.85
D5	0.63	0.89	0.89	0.88	0.74	0.88
Avg	0.71	0.88	0.89	0.87	0.79	0.87

We note that the results of DP shown in Table IV are not exactly the same as those reported in [26] because of the following reasons. First, we use different syntactic parsers (Minipar[4] is used in [26] while Stanford Parser is used in our experiments, both in DP and DPAP), which may produce slightly but not essentially different parsing results. Second, the authors of [26] mainly use frequency-based pruning method in their experiments. However, they also use a heuristic method to prune domain-specific candidate aspects, which is not always desirable as it trades recall for precision as reported in [26]. We do not use this heuristic method in our experiments and thus the recall of our DP implementation is much higher than that reported in [26]. The thresholds we use to prune infrequent aspects for DP are not exactly the same as those used in [26], which may also contribute to the differences.

[4]http://webdocs.cs.ualberta.ca/~lindek/minipar/

By our definition of general words, a total of 5,338 general words are found from WorldNet 2.0. Examples of general words are given in Section V. To illustrate the effectiveness of our proposed incorrect aspect identification method, we list some examples of non-aspects identified from the data sets by our ASP program in Table V. All words, except "format," "life" (such as in "battery life") and "function," in Table V are true incorrect product aspects, which explains the better precision of DPAP that prunes these non-aspects. Note that, general words such as "people," "thing," "reviews," "process," "purchase," "money" and "protection" in Table V are not infrequent candidate aspects in the data sets so that it is hard for frequency-based pruning method to get rid of them. Other general words in the table are infrequent candidate aspects, they can be pruned by both DPAP and DP.

TABLE V. EXAMPLES OF NON-ASPECTS IN THE DATA SETS. BOLD WORDS ARE WRONGLY EXTRACTED, AND UNDERLINED WORDS ARE TRUE INCORRECT ASPECTS THAT ARE NOT INFREQUENT IN THE DATA SETS.

person	friend	**life**	money	luxury
people	thing	line	company	care
summary	move	manager	headache	habit
succession	cable	matter	protection	health
brightness	method	title	chance	array
journey	kind	state	piece	pain
proof	vacation	process	scratch	aim
kitchen	**format**	purchase	benefit	charm
organization	place	stuff	fortune	band
construction	reviews	relation	**function**	model
upgrade	list	sort	difficulty	sheep

The words "format," "life" and "function" in the table are wrongly identified as incorrect aspects, which explains the slight drop of recall of DPAP shown in Table IV.

VII. CONCLUSION

This paper has shown that Answer Set Programming has provided an elegant, natural and concise implementation of the DP method for aspect extraction systems. The new approach significantly reduces the efforts to implement the extraction algorithm. It is also 10 times faster than a Java implementation on a collection of widely used data sets. We also proposed to augment the DP method with new knowledge for pruning aspects from the candidate ones. Experiments on 5 benchmark review data sets show that the augmented DP method outperforms the state-of-the-art DP method by a large margin. In future work, we plan to investigate more syntactic extraction rules, to refine knowledge for identifying incorrect aspects, and to conduct experiments on more data sets. We also plan to explore if other LP systems provide better performance (in terms of both modeling effectiveness and efficiency) than ASP.

ACKNOWLEDGMENT

We would like to thank Zhisheng Huang, Yaocheng Gui who contribute to this work. Yuanlin Zhang' work was partially supported by NSF grant IIS-1018031. Zhiqiang Gao's work was supported by the National Science Foundation of China under grant 61170165.

REFERENCES

[1] B. Liu, *Sentiment Analysis and Opinion Mining*, ser. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012.

[2] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '04, 2004, pp. 168–177.

[3] A.-M. Popescu and O. Etzioni, "Extracting product features and opinions from reviews," in *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT '05, 2005, pp. 339–346.

[4] G. Carenini, R. T. Ng, and E. Zwart, "Extracting knowledge from evaluative text," in *Proceedings of the 3rd international conference on Knowledge capture*, ser. K-CAP '05, 2005, pp. 11–18.

[5] L.-W. Ku, Y.-T. Liang, and H.-H. Chen, "Opinion extraction, summarization and tracking in news and blog corpora," in *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, 2006, pp. 100–107.

[6] N. Kobayashi, K. Inui, and Y. Matsumoto, "Extracting aspect-evaluation and aspect-of relations in opinion mining," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, June 2007, pp. 1065–1074.

[7] S. Blair-Goldensohn, T. Neylon, K. Hannan, G. A. Reis, R. Mcdonald, and J. Reynar, "Building a sentiment summarizer for local service reviews," in *Proceedings of WWW-2008 workshop on NLP in the Information Explosion Era*, 2008.

[8] Y. Wu, Q. Zhang, X. Huang, and L. Wu, "Phrase dependency parsing for opinion mining," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '09, 2009, pp. 1533–1541.

[9] T. Ma and X. Wan, "Opinion target extraction in chinese news comments," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, ser. COLING '10, 2010, pp. 782–790.

[10] J. Yu, Z.-J. Zha, M. Wang, and T.-S. Chua, "Aspect ranking: Identifying important product aspects from online consumer reviews," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11, 2011, pp. 1496–1505.

[11] K. Liu, L. Xu, and J. Zhao, "Opinion target extraction using word-based translation model," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, ser. EMNLP-CoNLL '12, 2012, pp. 1346–1356.

[12] N. Jakob and I. Gurevych, "Extracting opinion targets in a single- and cross-domain setting with conditional random fields," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '10, 2010, pp. 1035–1045.

[13] F. Li, C. Han, M. Huang, X. Zhu, Y.-J. Xia, S. Zhang, and H. Yu, "Structure-aware review mining and summarization," in *Proceedings of the 23rd International Conference on Computational Linguistics*, ser. COLING '10, 2010, pp. 653–661.

[14] Q. Mei, X. Ling, M. Wondra, H. Su, and C. Zhai, "Topic sentiment mixture: Modeling facets and opinions in weblogs," in *Proceedings of the 16th international conference on World Wide Web*, ser. WWW '07, 2007, pp. 171–180.

[15] I. Titov and R. McDonald, "A joint model of text and aspect ratings for sentiment summarization," in *Proceedings of ACL-08: HLT*, June 2008, pp. 308–316.

[16] C. Lin and Y. He, "Joint sentiment/topic model for sentiment analysis," in *Proceedings of the 18th ACM conference on Information and knowledge management*, ser. CIKM '09, 2009, pp. 375–384.

[17] Y. Lu, C. Zhai, and N. Sundaresan, "Rated aspect summarization of short comments," in *Proceedings of the 18th international conference on World wide web*, ser. WWW '09, 2009, pp. 131–140.

[18] H. Wang, Y. Lu, and C. Zhai, "Latent aspect rating analysis on review text data: A rating regression approach," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '10, 2010, pp. 783–792.

[19] S. Brody and N. Elhadad, "An unsupervised aspect-sentiment model for online reviews," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, ser. HLT '10, 2010, pp. 804–812.

[20] W. X. Zhao, J. Jiang, H. Yan, and X. Li, "Jointly modeling aspects and opinions with a maxent-lda hybrid," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '10, 2010, pp. 56–65.

[21] S. Moghaddam and M. Ester, "ILDA: interdependent lda model for learning latent aspects and their ratings from online product reviews," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, ser. SIGIR '11, 2011, pp. 665–674.

[22] C. Sauper, A. Haghighi, and R. Barzilay, "Content models with attitude," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11, 2011, pp. 350–358.

[23] Y. Jo and A. H. Oh, "Aspect and sentiment unification model for online review analysis," in *Proceedings of the fourth ACM international conference on Web search and data mining*, ser. WSDM '11, 2011, pp. 815–824.

[24] A. Mukherjee and B. Liu, "Aspect extraction through semi-supervised modeling," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ser. ACL '12, 2012, pp. 339–348.

[25] L. Zhuang, F. Jing, and X.-Y. Zhu, "Movie review mining and summarization," in *Proceedings of the 15th ACM international conference on Information and knowledge management*, ser. CIKM '06, 2006, pp. 43–50.

[26] G. Qiu, B. Liu, J. Bu, and C. Chen, "Opinion word expansion and target extraction through double propagation," *Computational Linguistics*, vol. 37, no. 1, pp. 9–27, Mar. 2011.

[27] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty, "Building watson: An overview of the DeepQA project," *AI Magazine*, vol. 31, no. 3, pp. 59–79, 2010.

[28] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995.

[29] S. Moghaddam and M. Ester, "Opinion digger: an unsupervised opinion miner from unstructured product reviews," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, ser. CIKM '10, 2010, pp. 1825–1828.

[30] Q. Su, X. Xu, H. Guo, Z. Guo, X. Wu, X. Zhang, B. Swen, and Z. Su, "Hidden sentiment association in chinese web opinion mining," in *WWW*, 2008, pp. 959–968.

[31] M. Gelfond and V. Lifschitz, "The stable model semantics for logic programming," in *Proceeding of the Fifth Logic Programming Symposium (ICLP/SLP)*, 1988, pp. 1070–1080.

[32] M. Gelfond, *Answer Sets*, ser. Handbook of Knowledge Representation. Elsevier, 2008.

[33] M. Davis, G. Logemann, and D. W. Loveland, "A machine program for theorem-proving," *Commun. ACM*, vol. 5, no. 7, pp. 394–397, 1962.

[34] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient sat solver," in *Proceedings of the 38th annual Design Automation Conference*, ser. DAC '01, 2001, pp. 530–535.

[35] M. Gebser, B. Kaufmann, and T. Schaub, "Conflict-driven answer set solving: From theory to practice," *Artif. Intell.*, vol. 187-188, pp. 52–89, Aug. 2012.

[36] W. Faber, G. Pfeifer, N. Leone, T. Dell'armi, and G. Ielpa, "Design and implementation of aggregate functions in the dlv system," *Theory Pract. Log. Program.*, vol. 8, no. 5-6, pp. 545–580, Nov. 2008.

[37] E. Erdem, J. Lee, and Y. Lierler, "Theory and practice of answer set programming – aaai12 tutorial," http://http://reu.uncc.edu/cise-reu-toolkit.

[38] G. Antoniou and A. Bikakis, "Dr-prolog: A system for defeasible reasoning with rules and ontologies on the semantic web," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 2, pp. 233–245, 2007.