Dependency Tree based Chinese Relation Extraction over Web Data

Shanshan Zheng, Jing Yang, Xin Lin*, JunZhong Gu
Computer Science and Technology Department
East China Normal University
Shanghai, China
xlin@cs.ecnu.edu.cn

Abstract—A new semi-supervised approach for Chinese relation extraction (RE) over constantly growing and edgeless web data is introduced in this paper. Existing semi-supervised approaches have the better improvement potential while lacking syntactic structure and semantic meaning of a sentence and unsuitable to loosely structured Chinese sentences. To follow their basic procedures as well as covering their remaining shortages, a dependency tree (DT) including both structure and semantic information is drawn in. Based on DTs, a new kind of pattern, called DT-based pattern, is proposed to extract new triples. Later patterns are optimized according to the characteristics of Chinese and typed dependency trees. Finally, extensive experiments show the higher precision and more efficiency of the proposed approach against DIPRE.

Keywords—semi-supervised, relation extraction, dependency tree, DT-based pattern

I. INTRODUCTION

For the constantly growing digitalized world, relation extraction (RE) from open areas becomes a hot topic. Against the domain-dependent and supervised traditional solutions like [1, 2, 3, 4], numerous existing scalable approaches such as [5, 6, 7, 8, 9, 10] are developed. They do not require large amounts of training data or prior knowledge for each relation. These approaches can be roughly classified into either unsupervised or semi-supervised category. Between them, the semi-supervised approaches inherit the advantages of supervised and unsupervised ones, and remedy their shortcomings. They do not need any annotated training corpus and can easily be extended to other new relations with good efficiency as discussed in [5, 6, 7, 8, 11].

Existing semi-supervised approaches still remain two unresolved problems. One is that most of them don’t take syntactic structure of a sentence into consideration. They only treat a sentence as a list of words and never study its syntax. The other is losing sight of the relationship of the words. Although some works such as [11] have add the semantic meaning into the semi-supervised RE, they still can’t address the issue well in Chinese documents. Chinese sentences are long, complex and loosely structured as described in [16], in which they may not have a very clear main clause, e.g. some sentences have two verbs without any conjunctions. Thus when dealing with such a complex sentence, they can not exactly pair the entities and the relation keywords, which will result in noisy patterns. For instance, in the triple occurrence in Figure 1, the correct match for the entities is that the second 美国(America) and the person 奥巴马(Obama) have a relationship of 总统(president), and the first 吸引(attract) has nothing to do with 奥巴马(Obama). However, the existing approaches may use the wrong pair of the first 美国(America) and 奥巴马(Obama), which will result in a wrong pattern.

To resolve these two problems, a new Chinese relation extraction (RE) approach using dependency tree (DT) over substantial and informal web data is proposed in this paper, which follows the procedures of the basic bootstrapping approaches. As we hypothesize that triples containing the same relation will share similar structure and context in sentences, DT, which can describe the synaptic structure and contents of a sentence, is a good choice.

Our approach will be triggered by a small set of triples without domain restriction and continually output a great

Figure 1. A Demo of Our Approach
quantity of extracted triples with high precision, as shown in Fig. 1. Thus, our main contributions can be concluded into the following points:

- A dependency trees based pattern is proposed. They are more than just a list of single words. They also contain the syntactic structure and the semantic context of the triples in the documents.
- DT-based patterns are optimized according to the characteristics of Chinese and grammatical relations in word pairs.
- DT can do more correct matches between entities and keywords not only in generating patterns but also in extracting new triples.
- To evaluate the approach, the new approach and DIPRE are implemented under Chinese condition and experiments in different relations are carried out. The result shows that our approach performs better on precision and recall.

The rest of this paper is organized as follows: Section II presents related work and Section III provides the preliminary about dependency tree. Section IV describes the procedures of our approach briefly. Section V describes the details of DT-base patterns in generation, optimization and matching. Finally, we present experimental results in Section VI and conclude in Section VII.

III. PRELIMINARY

Since the basic idea of our approach is adopting dependency tree to construct patterns, some knowledge about DT is reviewed in this section. [12] defines a dependency tree as a representation that denotes syntactic relations among words in a sentence as illustrated in Fig. 2(b). The nodes in the tree refer to the phrases in the sentence, and the edges connecting two neighboring nodes, one head node and one dependent node, indicate that they have some kind of dependency relation. As in Fig. 2(b), the words 奥巴马(Obama), the head, and 总统(president), the dependent, are related.

DT is generated based on the dependency grammar, which is based on the idea in [17] that the syntactic structure...
TABLE I. CHINESE GRAMMATICAL RELATIONS AND EXAMPLES

<table>
<thead>
<tr>
<th>Abbreviation Name</th>
<th>Detail Information</th>
<th>Abbreviated relation name (governor, dependent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>nn</td>
<td>noun compound modifier</td>
<td>美国总统 (American president)</td>
</tr>
<tr>
<td>nsbj</td>
<td>nominal subject</td>
<td>吸引,争夺 (attract, fight)</td>
</tr>
<tr>
<td>dobj</td>
<td>direct object</td>
<td>吸引,奥巴马 (attract, Obama)</td>
</tr>
<tr>
<td>advmod</td>
<td>adverbial modifier</td>
<td>也吸引 (also attract)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Abbreviation Name</th>
<th>Detail Information</th>
<th>Abbreviated relation name (governor, dependent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>advmod</td>
<td>adverbial modifier</td>
<td>也吸引 (also attract)</td>
</tr>
</tbody>
</table>

of a sentence consists of binary asymmetrical relations between the words of the sentence. Therefore, DTs contain the structures of the sentence and the semantic meanings of the words. What’s more, trees are not defined by specific words order, and thus related words of triples can be located everywhere. Thus using dependency trees, DT-based patterns can easily present the substructures of sentences and the closest context of triples.

To analyze the sentence structure in more detail, we choose the typed dependency, as shown in Fig. 2(b). It labels the dependencies with grammatical relations, and Table I explains some appeared ones in this paper.

In our implementation, the Stanford Parser, a statistical parser with a high accuracy developed by Stanford university described in [18], is applied to parse sentences. Their dependency tree is defined as follows:

- tree → tree, tree | unit
- unit → abbreviated relation name (governor, dependent)
- governor → phrase-location
- dependent → phrase-location

Here, abbreviated relation name refers to the grammatical relation between a governor and a dependent as Table I shows. phrase is a token in the sentence after part-of-speech (POS). location denotes the position of the phrase in the sentence, as phrases are marked from 1 by order. And the DT in Fig. 1 is the original text of Fig. 2(b).

IV. PROCEDURE OF OUR APPROACH

Since our approach is quite similar with the existing bootstrapping approaches, the procedure changes a little. Fig. 3 shows the whole process.

First, a small class of triples of certain relations, is treated as the seeds to initialize the procedures. The triples are formed as (e1, e2, keyword), such as (Obama, America, president) in Fig. 1, in which e1 and e2 refer to entities, and keyword indicates the relationship between e1 and e2. They have no restrictions with domain and the type of the named entities and keywords in one triple are also available to anyone.

Second, many occurrences of these given triples are gotten from the Internet. These sentences should exactly contain e1, e2 and keyword. As in Fig. 1, the triple occurrence actually includes 奥巴马(Obama), 美国(America) and 总统(president).

Third, the sentences are parsed to typed dependency trees like Fig. 2(b), which are then used to generate DT-based patterns. Later the patterns are filtered and improved. In Fig. 1, a pattern "nn(总统, ns2), nn(nr1, 总统), dobj(吸引, nr1)" is created, where ns2 means that e2 美国(America) is a name of place marked by ns, and nr1 means that e1 奥巴马(Obama) is a name of person labeled by nr. They both represent the types of the named entities in the triple. More detail information is described in Section 5.

Next, sentences containing all the phrases in the DT-based patterns, excluding the types of the named entities such as nr1 and ns2, are collected as pattern occurrences. In the demo of Fig. 1, the pattern occurrences should include 总统(president) and 吸引(attract), and thus the example sentence in the figure is selected.

Finally, the new triples are extracted when the dependency tree of a pattern occurrence matches the DT-based pattern exactly. In part C of Section V, a further explanation about pattern match and new triple extraction is given.

On the whole, our approach is bootstrapping. The new generated triples after one cycle are treated as seeds in the next iteration, and the program will continue running until no new triples are extracted. What’s more, the program is domain-independent and doesn’t need any training corpus. It can easily transfer to many other relations.

V. DT-BASED PATTERN

A. Pattern Generation

With the typed dependency representations, the patterns are available. Our DT-based pattern can be defined as follows:

Figure 3. Whole procedures of our approach
used to pathE1ToE2Set e1

appended to indicate represents the tag of the first entity patterns are as follows:

Therefore, the smallest sub-tree including the seed is used to another, the more close relationship between them.

sentence. We assume that the shorter path from a word to related context of the triple and the syntactic structure of the dependency tree. Therefore, the pattern contains the closely

storing the cost from a node to another node. Then find the shortest paths between e1 and e2 and between e1 and keyword. Next, construct a shorter path linking e1, e2 and keyword according to the shortest path. Finally, the path is extended, and the pattern is created by replacing the entities with (e1.tag)1 or (e2.tag)2 and clearing the locations. For the case in Fig. 1, the pattern generation is shown in Fig. 4. The pseudo code about how to generate the pattern is shown as follows:

<table>
<thead>
<tr>
<th>Pattern Extractor:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> the seed e1, e2 and keyword, the occurrence S used to extract the pattern, the type of e1 e1.tag and the type of e2 e2.tag</td>
</tr>
<tr>
<td><strong>Output:</strong> the pattern P</td>
</tr>
<tr>
<td>1: segment ( \leftarrow ) POS(S)</td>
</tr>
<tr>
<td>2: DT ( \leftarrow ) generate dependency tree(S)</td>
</tr>
<tr>
<td>3: e1Pset, e2PSet, keywordPSet ( \leftarrow ) get position(e1, e2, keyword, segment)</td>
</tr>
<tr>
<td>4: costVector ( \leftarrow ) initialize cost(DT)</td>
</tr>
<tr>
<td>5: for each eIP in e1Pset</td>
</tr>
<tr>
<td>6: for each e2P in e2PSet</td>
</tr>
<tr>
<td>7: pathE1ToE2Set ( \leftarrow ) get shortest path(e1P, e2P, costVector)</td>
</tr>
<tr>
<td>8: endfor</td>
</tr>
<tr>
<td>9: for each keywordP in keywordPSet</td>
</tr>
<tr>
<td>10: pathE1ToKSet ( \leftarrow ) get shortest path(e1P, keywordP, costVector)</td>
</tr>
<tr>
<td>11: endfor</td>
</tr>
<tr>
<td>12: endfor</td>
</tr>
<tr>
<td>13: shortestPath ( \leftarrow ) get the shortest path(pathE1ToE2Set, pathE1ToKSet)</td>
</tr>
<tr>
<td>14: path ( \leftarrow ) extend the node has connection with e1, e2 and keyword (shortestPath, costVector)</td>
</tr>
<tr>
<td>15: P ( \leftarrow ) translate to pattern form(path, DT, e1.tag, e2.tag)</td>
</tr>
<tr>
<td>16: return P</td>
</tr>
</tbody>
</table>

B. Pattern Optimization

According to the aforementioned procedure, many irrelevant patterns may be generated. Therefore, it is necessary to do optimization to improve the patterns’ quality.

In the algorithm above, the DT-based pattern can handle all the 45 grammatical relations in [19]. However, they are not equally important. As Table I shows, in advmod(A, B), B is only used to modify something, not important in Chinese grammar. However, in nsubj(A, B), A must be a host of what happened and B must be the action or something to describe situation, both representing the characteristics of a sentence. Therefore, different solutions should be used in different kinds of abbreviated relation names.

On the other hand, [18] shows that the grammatical relations of typed dependencies stand in a hierarchy. In [18], the most generic grammatical relation, dependent(dep), will be used when a more accurate relation in the hierarchy does not exist or cannot be retrieved by the system. For example in [20], dep can be classified to auxiliary, argument (arg), or modifier. arg can be further grouped into either subject (subj) or complement (comp). The deeper the relations are in the hierarchy, the more important they are.

\[ \ldots, \text{assmod(女足-5, 美国-3), nsubj(吸引-10, 女足-5)}, \ldots, \text{dep(吸引-10, 吸引-19), nn(总统-22, 美国-21), nn(奥巴马-23, 总统-22), dobj(吸引-19, 奥巴马-23)} \]

Get shortest path

\[ 23 \rightarrow 22 \rightarrow 21 \]

Extend the path

\[ 19 \rightarrow 23 \rightarrow 22 \rightarrow 21 \]

Get DT path

\[ \text{nn(总统-22, 美国-21), nn(奥巴马-23, 总统-22), dobj(吸引-19, 奥巴马-23)} \]

Clear location and replace entities

\[ \text{nn(总统, ns2), nn(nrl1, 总统), dobj(吸引, nrl1)} \]

Figure 4. Example of pattern generation

\[ \text{pattern} \rightarrow \text{pattern, pattern | element} \]

\[ \text{element} \rightarrow \text{abbreviated_relation_name(governor, dependent)} \]

\[ \text{governor} \rightarrow \text{word | (e1.tag)1 | (e2.tag)2} \]

\[ \text{dependent} \rightarrow \text{word | (e1.tag)1 | (e2.tag)2} \]

Here, (e1.tag)1 refers to a string like nrl1. e1.tag represents the tag of the first entity e1, and then l is appended to indicate e1; Similarly, in (e2.tag)2, e2.tag refers to the tag of the second entity e2, to which 2 is appended to indicate e2. word denotes a token in the sentence after POS except e1 and e2.

In this paper, a pattern is a substructure of the dependency tree. Therefore, the pattern contains the closely related context of the triple and the syntactic structure of the sentence. We assume that the shorter path from a word to another, the more close relationship between them. Therefore, the smallest sub-tree including the seed is used to locate the entities and the keyword, and the requirements of patterns are as follows:

- The pattern is a sub tree of the dependency tree containing e1, e2 and keyword.
- Based on the smallest sub tree, which just contains the shortest paths among e1, e2 and keyword, the sub tree extends one dependent or head out from the entities’ or keyword’s location.

Following these conditions, when generating the DT-based pattern, we firstly get all the locations of e1, e2 and keyword, and then transfer the dependency tree to a matrix storing the cost from a node to another node. Then find the
What’s more, in Chinese sentences, subjects, predicates and objects are more important than adverbials, attributives and complements. Since the latter three are mainly modifiers, they are ignored in our pattern recognition. Then the second patterns production rules mentioned in part A of Section V is optimized as follows:

\[ \text{element} \rightarrow \text{selected_relation_name(governor, dependent)} \]

Here, we mainly focus on the \text{selected_relation_name} which is the grammatical relations only involved with subject, predicate and object such as nn, nsubj and dobj in Table I. Others like advmod is beyond the scope of this paper.

Therefore, patterns are improved by the following algorithm:

```
PATTERN OPTIMIZATION:
Input: the pattern set \( p\)Set and selection relation name SRS\Set Output: the optimized pattern OP\Set
1: for each \( p \) in \( p\)Set
2: eSet = get all the elements (\( p \))
3: for each \( e \) in eSet
4: if the relation name of \( e \) is not in SRS\Set
5: remove \( e \)
6: endfor
7: connected \( \leftarrow \) check the connection of elements (e\Set)
8: if connected is true
9: OP \( \leftarrow \) construct the pattern(e\Set)
10: add the OP to OP\Set
```

C. Pattern Matching

After getting the refined patterns, the next critical part is matching the patterns with pattern occurrences to extract new triples. In this step, the following two conditions should be satisfied:

- All the elements in the pattern are matched by the units of the typed dependency tree. A unit matches an element only if the unit of occurrence matches all the characters, except \( (e1.tag)1 \) and \( (e2.tag)2 \) in the element of the pattern.
- Unless the entities in the extracted triples exactly match the types in the pattern, will they be extracted to be a new triple, together with the keyword. The first entity’s type should be the same with \( e1.tag \), and the second entity’s type should also be the same with the \( e2.tag \).

Therefore, the pattern match pseudo code is as follows:

```
PATENT MATCHER:
Input: the pattern \( p \) which result in the sentence and the sentence \( S \)
Output: the extracted triple \( T \)
1: DT \( \leftarrow \) parse the sentence to DT(\( S \))
2: for each element in \( P \)
3: if element be matched by one element \( e \) in DT
4: e\Set \( \leftarrow \) add \( e \) to the match set e\Set
5: else return Null
6: endfor
7: T \( \leftarrow \) select the candidate triple(e\Set, \( p \))
8: types \( \leftarrow \) get the entity types in pattern(\( p \))
9: if \( T \) matches types
10: return \( T \)
11: else return Null
```

For the case in Fig. 1, the new triple (梅德韦杰夫, 俄罗斯, 总统) ((Medvedev, Russia, president)) is extracted with the steps in Fig. 5. It can be easily find that units in bold match the pattern, and 俄罗斯(Russia) and 梅德韦杰夫(Medvedev) become the candidate. In addition, 俄罗斯(Russia) is a place tagged by ns and 梅德韦杰夫(Medvedev) is a person labeled by nr, so the triple (梅德韦杰夫, 俄罗斯, 总统) ((Medvedev, Russia, president)) is correctly exacted.

VI. EXPERIMENT RESULTS

Precision, recall and the combined recall-precision score F-measure are commonly used performance criteria to evaluate Information Extraction methods. However, our approach is applied on the web. Since the amount of documents is beyond number, it is infeasible to calculate recall and F-measure. Yet, to get a more detail comparison result, we choose the amount of the new extracted triples as a substitution, which can reflect the recall measure.

A. Experiment Setting

In this paper, the performance between DIPRE, DT-based pattern and optimized DT-based pattern are compared.
All methods are initialized with the same predefined relations, and continuing running four iterations. In the experiments, several different relations in different domain are implemented and to each relation, four seeds are assigned and four cycles are completed.

Three frequently occurring relations, **总统**(president), **校长**(schoolmaster) and **首都**(capital), are chosen to assess our approach. They represent three of the seven syntactic relation classes in the guidelines of automatic content extraction, general affiliation, Org-affiliation and physical[21].

Later, DTP and O- DTP is used in the tables and figures to represent DT-based pattern and optimized DT-based pattern for short.

### B. Precision Evaluation

It can be seen from Table II that precisions of DT-based pattern are all around 90%, and the highest is about 98.94%, while the highest one of DIPRE is only 84.47% and most of its precisions are less than 70%. Thus, DT-based pattern gets better precision than DIPRE in all three relations.

Comparing to DT-based pattern, higher precisions are achieved by the improved patterns. The only lower accuracy also reaches a really high value, 98.75%, only about 0.2% lower, which results from less patterns after optimization. Therefore, less pattern occurrences are collected, and less triples and less correct triples are extracted. Smaller numerator and denominator cause the little lower precision. Although not in all relations the optimized pattern outperforms, it could still be claimed that it does better work.

As shown in Fig. 6, our approaches do better on complex sentences because syntactic structure and semantic meaning are considered. The precisions of DT-based patterns and improved patterns are almost 10% or more higher than DIPRE. Improved patterns gets the highest in the whole.

### C. Recall Evaluation

From Table II, the extracted triples over three relations using DT-based pattern are 5834, 1746 and 2176, all much bigger than the ones using DIPRE, whose largest amount is 472. The huge gap comes from the numbers of words in the pattern. The patterns in DIPRE retain all the words in the sentences except the entities in the triples, resulting in very few pattern occurrences, while DT-based patterns only contain the closely context and much more pattern occurrences are found. Since recall-measure is defined as (1), DT-based patterns gets higher recalls than DIPRE.

\[
\text{recall} = \frac{\text{amount of correct triples}}{\text{amount of correct triples in corpus}} \tag{1}
\]

What’s more, optimized DT-based pattern gets more triples over most relations. The exception in Table II is capital. To this phenomenon, two factors are responsible. One is the time dynamicity of the web sources. The other is reduced amount of the patterns through the optimization of patterns, which caused the reduced number of occurrences. Therefore, the contributes of optimization can’t be concluded only in the recall.

### D. Pattern Comparison

In our approach, the DT-Based patterns generated by the algorithm in part A of Section V is refined in pattern optimization. Table III presents the effectiveness of optimization in term of efficiency, avg(len) and precision. Here, efficiency refers to the percentage of the patterns which can extract new triples in proportion to the total patterns, avg(len) represents the average character length of pattern, and precision is the average precision of the patterns, calculated as

### Table II. Precision Comparison of DIPRE, DTP and ODTP over three relation

<table>
<thead>
<tr>
<th>Relation</th>
<th>Approach</th>
<th>Triple Number</th>
<th>Pattern Number</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>President</td>
<td>DIPRE</td>
<td>264</td>
<td>25</td>
<td>0.8447</td>
</tr>
<tr>
<td></td>
<td>DTP</td>
<td>5834</td>
<td>237</td>
<td>0.9357</td>
</tr>
<tr>
<td></td>
<td>O- DTP</td>
<td>6307</td>
<td>224</td>
<td>0.9629</td>
</tr>
<tr>
<td>Schoolmaster</td>
<td>DIPRE</td>
<td>472</td>
<td>70</td>
<td>0.6758</td>
</tr>
<tr>
<td></td>
<td>DTP</td>
<td>1746</td>
<td>617</td>
<td>0.8751</td>
</tr>
<tr>
<td></td>
<td>O- DTP</td>
<td>1834</td>
<td>566</td>
<td>0.8833</td>
</tr>
<tr>
<td>Capital</td>
<td>DIPRE</td>
<td>196</td>
<td>25</td>
<td>0.6378</td>
</tr>
<tr>
<td></td>
<td>DTP</td>
<td>2176</td>
<td>605</td>
<td>0.9894</td>
</tr>
<tr>
<td></td>
<td>O- DTP</td>
<td>2083</td>
<td>590</td>
<td>0.9875</td>
</tr>
</tbody>
</table>

### Table III. Effectiveness of Optimization

<table>
<thead>
<tr>
<th>Relation</th>
<th>Approach</th>
<th>Efficiency</th>
<th>Avg(len)</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>President</td>
<td>DTP</td>
<td>0.3385</td>
<td>16.73</td>
<td>0.9353</td>
</tr>
<tr>
<td></td>
<td>O- DTP</td>
<td>0.4273</td>
<td>15.20</td>
<td>0.9356</td>
</tr>
<tr>
<td>Schoolmaster</td>
<td>DTP</td>
<td>0.5575</td>
<td>17.51</td>
<td>0.9408</td>
</tr>
<tr>
<td></td>
<td>O- DTP</td>
<td>0.6429</td>
<td>16.96</td>
<td>0.9260</td>
</tr>
<tr>
<td>Capital</td>
<td>DTP</td>
<td>0.7194</td>
<td>19.04</td>
<td>0.9797</td>
</tr>
<tr>
<td></td>
<td>O- DTP</td>
<td>0.7268</td>
<td>19.35</td>
<td>0.9907</td>
</tr>
</tbody>
</table>

Figure 6. Precision of DIPRE, DT-based pattern and weighted DT-based pattern of three relations keywords on complex sentence.
\[
\text{Precision} = \frac{\sum P_i}{n}, i \in [0, n]
\] (2)

In (2), \( P_i \) is the precision of pattern \( i \) which is computed as

\[
P_i = \frac{\text{amount of correct triples generated by pattern } i}{\text{total amount of triples generated by pattern } i}.
\] (3)

For all the relations in Table III, the efficiency is improved through optimization as patterns containing grammar relations not in selected sets are filtered. These patterns include too much modifiers to match sentences. The filter also results in shorter patterns. In Table III, in most cases, optimized pattern have shortened average lengths and higher precisions. The exception of schoolmaster is caused by the mistakes of part-of-speech. In the new instances extracted by optimized patterns, we discovery more misunderstandings of entities.

Overall, Table III proves that the optimization is efficient. And considering the better precision and close number of extracted triples, it is believed that the approach with optimization goes a little better.

VII. CONCLUSION

In this paper, we propose a new approach of relation extraction in Chinese using dependency trees to address the problems of syntactic structure and semantic meaning of sentences. For extraction, new patterns, DT-based patterns, are introduced and patterns are refined later. Comparing to DIPRE, our approach achieves better performance on precision and recall, and the optimized pattern are more efficient.

In the future, we can do more researches on dependency tree and Chinese language, raising the precision of DT and the parsing speed, and assigning more suitable value to grammatical relations according to their importance in sentences in order to achieve better results.

ACKNOWLEDGMENT

This work is supported by the Opening Project of Shanghai Key Laboratory of Integrate Administration Technologies for Information Security, and a grant from the National High Technology Research and Development Program of China (863 Program) (No. 2011AA01A107), the grant from the Shanghai Science and Technology Foundation (No. 10dz1500103 and No. 11530700300).

REFERENCES