Journal of Web Engineering, Vol. 12, No.5 (2013) 363-382

A STRUCTURAL APPROACH TO EXTRACTING CHINESE POSITION RELATIONS FROM WEB PAGES

PEIQUAN JIN JIA YANG

University of Science and Technology of China, China

jpq@ustc.edu.cn yangjia@mail.ustc.edu.cn

JIE ZHAO

Anhui University, China

zj_teacher@126.com

YANHONG LIU

University of Science and Technology of China, China

keen0088@gmail.com

Received November 8, 2012
Revised March 22, 2013

The use of position relations, which refer to the position of people in an organization, can serve for enterprises as a significant competitive intelligence method. The rapid growth of the data volume in the Web brings new opportunities for us to extract position relations of interest from the Web. In this paper, we propose a new algorithm to extract position relations from the Web. Our algorithm is based on the structural feature of position relations in the Web, i.e., a position relation is usually presented in Web pages as a table or a list. In order to define the structural feature of Web content, we first introduce a structural coefficient for each Web page, which is then used to generate structural file segments for Web pages. A structural file segment consists of all candidates of position relations having a similar structure. After that, we employ a pattern-matching method to extract position relations from the structural file segments. Finally, we conduct experiments on a real data set containing 6028 Chinese Web pages gathered by the Baidu search engine, and evaluate precision and recall of our approach. The experimental results confirm that our algorithm has a precision over 96% and a recall over 87%.

Key words: Position Relation, Relation Extraction, Structural File Segment

1 Introduction

A position relation describes the fact that a person holds a position in a specific organization. A position relation typically contains three elements, which can be formalized as a triple $\{O, P, R\}$, where O, P and R stand for organization name, position name, and person name, respectively. Position relation extraction aims at obtaining such position triples from natural language texts or Web pages.

The position relation can serve as an important intelligence method in business competition. If a company can reveal the position relation of the management team or the R&D department of its

competitors, it will be helpful to enhance the core competence of the company, especially useful for the global competition in the information age.

Previous work on relation extraction was based on named entity recognition. They first recognized the named entities in a Web page and then used either a model-based approach [1-5] or a kernel-based approach [6-10] to extract relations among entities. These methods mostly depend on the effectiveness of the algorithms on named entity recognition. Due to the low recall of organization entity recognition [11, 12], traditional relation extraction approaches are not suitable for position relation extraction. There are also some Web databases or tools related to position relation extraction, such as Wikipedia and LexisNexis (http://www.lexisnexis.com). Wikipedia usually provides ownership information in its company profile pages. LexisNexis, which is a well-known commercial information service for legal, economic, and media information, can gather information from different sources, e.g., news, company financial reports, accounting literature, and company registration databases, and produce structural descriptions about companies. The only element concerning position relation in the description, called *key executive*, usually represents the CEO of a company, which can be regarded as one type of position relations. However, both Wikipedia and LexisNexis can only extract specific position relations from the Web, i.e., ownership or executive, which is not enough for competitive intelligence analysis. For example, queries like "*Is Tom an engineer at Microsoft*?" are difficult to be answered by those systems, as they can not recognize the position relation "*engineer*".

In this paper, we propose a novel approach to extract position relations from Web pages. A preliminary version of our work appears in 11th ACM International Workshop on Web Information and Data Management [13]. However, we have made significant extension both on the key techniques and experiments. Our approach takes advantage of the structural feature of position relations in Web pages and overcomes the low-recall problem of traditional organization-entity recognition algorithms. Careful observation reveals that a lot of position relation instances in Web pages appear in a specific structural segment, e.g., a list or a table. These segments have the following common features:

(1) Each relation instance appears visually in and only in one line of the Web browser.

(2) Each relation instance contains at least one special symbol separating the elements in the position relation. Typical symbols include "blank", "colon", and so on. We denote these symbols as separators in the following text.

(3) Most relation instances have a similar structure, i.e., the positions of the same element in different position relation instances are identical. Hence, we can use the structural feature to figure out position relations even though we do not recognize the organization entity, in case that we have got the common structure of position relations.

(4) The three elements of a position relation instance usually have only few variations. This makes the extraction task much easier, because separators can be seen as the natural boundaries of the position relation elements.

The main contributions of the paper can be summarized as follows:

(1) We present a structure-based approach to extract the position relations from Web pages. Compared with the previous named-entity-based approaches, our policy has less dependency

on the fundamental algorithms for named entity recognition. In particular, it can still extract the right position relation, even though the organizational element has not been recognized.

(2) We introduce two new ideas, namely the structural coefficient and structural file segment, to describe the structural features of a Web page. Our experimental results show that they can capture the Web pages' structural features effectively.

(3) We conduct experiments on a real data set containing 6028 Chinese Web pages gathered by the Baidu search engine, and evaluate the precision and recall ratios of our proposed approach. The experimental results show that our algorithm has an excellent precision of 96.10% and a recall ratio of 87.16%, on average.

The remainder of the paper is organized as follows. Section 2 discusses related work. In Section 3, we introduce the general framework of our approach to extract position relations from Web pages. Section 4 outlines the extraction of structural file segments, which forms the foundation of our method. Section 5 describes the detailed algorithms of position relations extraction. Section 6 reports the experimental results, and conclusions and future work are given in Section 7.

2 Related Work

Relation extraction from the Web usually employs a model-based approach [1, 3-5]. The model-based approach generates models through seed relations and then creates more relations by computing similarities between models and candidate relation instances. During the similarity computation, entity types in the relation instances must first be tagged. If the type of an entity does not meet the requirements, the degree of similarity will be set to 0. This traditional model-based approach does not suit Chinese position relation extraction because of the low recall of Chinese organization entity recognition. Such a recognition method usually adopts a tail-word-based method [11, 12]. A formal Chinese organization name usually ends with a special word such as "公司(corporation)" or "集团(group)". We call these special words organization feature words (OFW). A tail-word-based method regards OFW as symbols of the rear boundary of Chinese organization entities. During the recognition process, an organization feature word is found at first, and then a forward match method is applied to confirm the head boundary of a Chinese organization entity. However, a lot of Chinese organization names appear in short form in Web pages. These short-form names usually do not include organization feature words and can not be recognized by tail-word-based methods. For example "百度在线网络技术有限公司(Baidu online network technology Co. Ltd.)" often appears in the form of "百度(Baidu)". Moreover, selection of seed relations is also a complicated task for traditional model-based relation extraction approaches.

The employment relation extraction defined by ACE (Automatic Content Extraction) is somewhat similar to our task, but it just needs to detect whether a person works for an organization and pays little attention to positional information. In other words, the result of an *Employment* relation extraction is a binary relation <*Organization*, *Person*> rather than a triple.

Huang et al. regards information about positions as a complement to employment relations [8]. However, they do not eliminate the negative recognition effect of Chinese organization entities. Furthermore, the details of extracting position information are not presented in their paper.

Compared with the traditional model-based approach, our approach has three distinct properties:

(1) It does not need seed relations to generate models.

(2) It does not depend on recognition of Chinese organization entities.

(3) The result of position relation extraction is a triple set $\{<O_i, P_i, R_i>\}$ rather than a binary set $\{<O_i, R_i>\}$.

3 Framework of Extracting Position Relations

The framework of position relation extraction is shown in Figure 1. It consists of two key parts: extraction of structural file segments and relations extraction.

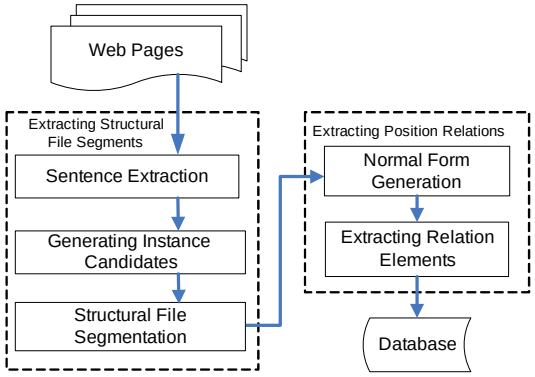

Figure 1 Framework of extracting position relations from Web pages

3.1. Extracting Structural File Segments

This module takes Web pages as input and finally generates a structural file segment for each Web page. A structural file segment consists of all the candidates of position relations having a similar structure. We will use a structural coefficient and develop several algorithms to achieve this goal (see Section 4).

3.2. Extracting Position Relations

In this module, we process the structural file segments generated by the previous module and return the final position relation triples. It employs a two-stage method, in which at first the normal form for position relations is defined and then a pattern-matching method is used to extract the position relations (see Section 5).

4 Extracting Structural File Segments

4.1. Structural Coefficient and Structural File Segments

Our extraction algorithm for position relations is basically founded on two new concepts, i.e., *structural coefficient* and *structural file segment*. Before we present the detailed algorithm, we give at first the formal definitions of these concepts.

Definition 1 (*Position Relation Instance Candidate*) If a sentence contains a person name, a position name and at least one separator, we call it a *position relation instance candidate*, or *instance candidate* for short. ■

An instance candidate has the form like "f_1—f_2—...—f_i—...—f_n". Here, f_i is the *i-th* word phrase and "—" denotes a separator. Table 1 shows some general separator symbols used in Web pages, which are also employed in this paper to generate instance candidates. In addition, we call each f_i a *field* of an instance candidate. Moreover, the number of all the fields in an instance candidate is limited by a certain value called *max field count (MFC)*. This policy ensures that, if a sentence contains more than *MFC* fields, it will not be considered as a candidate of position relations. Because the result of position relation extraction is a triple set $\{<O_i, P_i, R_i>\}$ and these three elements may occur in different fields of an instance candidate, *MFC* is not less than 3. The impact of *MFC* on our approach will be discussed in the experimental section.

Table 1. Some general symbols to separate word phrases in Web pages

Symbol	Meaning	Example
	blank	杨宁(*Yang Ning*) 空中网总裁(*the president of Kong Zhong Website*)
:	colon	千橡集团总裁(the president of Qian Xiang Group)：陈一舟(Chen Yizhou)
—	dash	大贺集团董事长(the board chairman of Da He Group)—贺超兵(He Chaobin)
<td> ...</td>	column tag	**<td>**中华广告网董事长(the board chairman of Zhong Hua Advertisement Website)**</td><td>**姜杉(Jiang Bing)**</td>**
(...)	parenthesis	雷军(Lei Jun)（金山总裁(the president of Jin Shan)）

Definition 2 (*File Segment*) The *File Segment* of a Web page is the set of all instance candidates in the page. ■

Definition 3 (*Structural Coefficient*) The *Structural Coefficient* of a file segment defines the structural level of the file segment. It is evaluated using the following formula:

$$sc = \frac{{p_m}^2}{{p_m}^2 + c\sum_{\substack{i=2,\\ i\neq m}}^{mfc} t(p_i) \sum_{\substack{i=2,\\ i\neq m}}^{mfc} p_i}$$

where (1) p_i is the number of such instance candidates that contain exactly i fields.

(2) p_m is the maximum of all p_i.

(3) *mfc* is a constant indicating the maximum possible number of fields in any instance candidate.

(4) $t(p_i)$ is 0 if $p_i = 0$ and it is 1 if $p_i > 0$. $\sum t(p_i)$ counts the sub-sets of instance candidates, which are partitioned according to the count of fields contained in them. Particularly, each sub-set of instance candidates contains a certain but different number of fields. For instance, if a file segment has 100 instance candidates, 10 of which contain s fields and the rest all contain w fields, where $s \neq w$, then $\sum t(p_i)$ is 2.

(5) c is a negative factor which reflects the variance impact of field counts on the structural level. ■

The structural coefficient reflects the structural level of a file segment. According to this definition, the number of instance candidates having maximum (m) fields reflects the structural feature of the file segment. Moreover, if the instance candidates have different field counts, the structural feature of the file segment will be influenced. We introduce $\sum t(p_i)$ in the formula to capture such an influence. $\sum t(p_i)$ exerts influence on c. The greater $\sum t(p_i)$, the larger is the negative influence of c.

Definition 4 (*Structural File Segment*) If the structural coefficient of a file segment is over a given threshold α, we call this file segment a *Structural File Segment.* ■

4.2. Sentence Extraction

As Figure 1 shows, the first step to extract structural file segments is the sentence extraction. Like traditional relation extraction, we also attempt to extract position relations from a single sentence. However, our approach to extract sentences is different from others due to our special consideration of the structural feature in Web pages. We extract sentences from the structural parts in a Web page, e.g. from a list or a table in a Web page, rather than from a common paragraph. Those sentences are usually separated by some HTML tags assigning each sentence into a separate line of the Web browser. From this viewpoint, we regard a Web page as a long string and segment it into sentences according to the special HTML tags. A set of sentences are obtained after segmentation.

4.3. Generating Instance Candidates

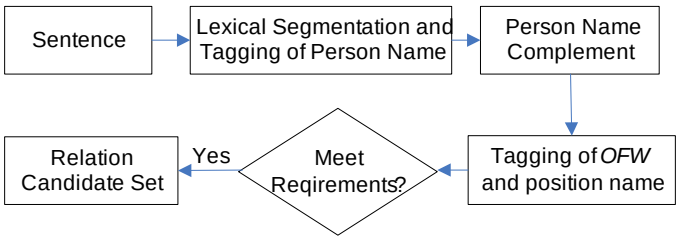

Figure 2. Generating instance candidates

As shown in **Definition 1**, an instance candidate does not ask for an explicit organization name because of the low recall of organization entity recognition. In Section 4.4, we will identify whether there is an organization name in an instance candidate. Moreover, an instance candidate must contain

at least one separator because it serves to separate the three elements of position relations and helps us to ascertain the boundaries of the elements. Different separators serve the same purpose and will not influence our extraction result, so all those separators shown in Table 1 are converted to dashes ("—") by our algorithm. Figure 2 shows the procedure of getting instance candidates from a sentence.

(1) *Tagging of person names*

Here we use ICTCLAS [14] as tool for lexical segmentation and tagging of person names. ICTCLAS is a popular tool to recognize named entities for Chinese documents, and has been used in many areas including Web information extraction [15] and information retrieval [16]. However, the ICICLAS tool can not recognize some special person names in Web pages, such as "杨(Yang) 皓(Hao)", where there is a blank character between surname and given name. In this paper, we add two rules to ICTCLAS to tackle with these special cases and denote this process as person name complement.

Rule 1: After lexical segmentation and POS tagging, if there is a substring like "A/*nr1* # B/*x*" or "A/*nr1* B/*x* —", "A" and "B" will be combined and tagged "/*r*" to obtain "AB/*r*".

Rule 2: After lexical segmentation and POS tagging, if there is a substring like "A/*nr1* BC/*x*—" or "A/*nr1* B/*x* C/*x* —", "A", "B" and "C" will be combined and tagged "/***r***" to obtain "ABC/***r***".

In these rules, "A", "B", or "C" denote a single Chinese character, respectively, tag "*nr1*" denotes that "A" is a Chinese surname. Tag "*x*" represents any POS tag. Symbol "#" denotes a blank character, while "—" represents separators described above. Finally, tag "*r*" denotes a complete Chinese person name.

For example, substring "杨(Yang)/*nr1* # 皓(Hao)/*ng*" will be converted to " 杨皓(Yang Hao)/*r*" according to **Rule 1** and substring "蒋(Jiang)/*nr1* 天龙(Tianlong)/*nz* —" will be converted to "蒋天龙(Jiang Tianlong)/***r*** —" according to **Rule 2**.

(2) *Tagging of position names*

Position name tagging is conducted using a position dictionary which is manually constructed. Note that the dictionary only contains simplified position names. A simplified position name only contains core words of a complete position name. For example, "总裁(president)" is a simplified position name, while "副总裁(vice president)" is not a simplified position name. The tag of a position name is "/***p***".

(3) *Tagging of potential organization names*

A potential organization name in a sentence is tagged by the *OFW* (*organization feature word*). If a sentence contains a person name, a position name, and an *OFW* simultaneously and these elements appear in a specific structure, then an organization name is probably in the *OFW* position. We first assume an organization name in the *OFW* position and filter out the sentences which contain an illegal organization name in the subsequent stages. In this paper, we tag the two most frequent *OFWs*: "公司(corporation)" and "集团(group)", both with the tag "/***o***". Table 2 shows a summary of the tagging symbols used by our algorithm, and Figure 3 shows an example of an instance candidate generated.

Table 2. Tagging symbols in instance candidates

Symbol	Meaning
/***r***	a person name
/p	a position name
/o	an *OFW* (organization feature word) tag, i.e., a potential organization name

Sentence: 天极公司总裁(the president of Tian Ji corporation)
一李志高(Li Zhigao)

Instance Candidate: 天极(Tian Ji) 公司(corporation)***/o***
总裁(president)***/p*** 一 李志高(Li Zhigao)***/r***

Figure 3. An example of instance candidate

4.4. Structural File Segmentation

A structural file segment is obtained using a structural coefficient, which we have already introduced to reflect the structural level of a file segment. If the structural coefficient of a file segment is equal or above a certain threshold α, it is regarded as a structural file segment.

In addition, a structural file segment also has the following properties:

(1) It contains at least two instance candidates. If a file segment only contains one instance candidate, it is almost impossible that this instance candidate is extracted from a structural part of a Web page, and therefore little confidence exists that the instance candidate will contain a position relation.

(2) It contains at least one instance candidate which has an *OFW* tag. If none of the instance candidates contains an *OFW* tag, we will not be able to infer the positions of organization names in the instance candidates because we have no referential information.

Here, we do not consider the element position consistency of a position relation in instance candidates, so some ambiguous instance candidates nay occur in structural file segments. They will be filtered out in the stage of element extraction (see Section 5.2).

5 Extracting Position Relations

After having identified the structural file segments of Web pages, we first generate a *normal form* for each structural file segment. The normal form indicates the element position information of position relations in the instance candidates. Then, we will extract the elements of position relations from the instance candidates. And finally, we need to further evaluate the correctness of each element to get the

final position triples. The right dashed rectangle in Figure 1 shows the two main steps of position relation extraction.

5.1. Normal Form Generation

The normal form generated from a structural file segment is based on the concept of *position relation schema*.

Definition 5 (*Position Relation Schema*) The *position relation schema* refers to the general structure of an instance candidate. It is defined as follows:

$$S(I) = \left\{ w \mid w =< s_I^1, ..., s_I^i, ..., s_I^n > \wedge s_I^i \in \{O, P, R, N, OP, OR, PR, OPR\} \right\}$$

Where I is an instance candidate, w is a sequential list, each of whose elements is a flag indicating the field type, which is taken from the eight field types {*O, P, R, N, OP, OR, PR, OPR*} ■

"*O*", "*P*" "*R*" and "*N*" are the basic four field types, representing an organization name, a position name, a person name and a name other than these three types, respectively. Unions of these four basic field types construct another four field types "*OP*", "*OR*", "*PR*" and "*OPR*". The field types are computed as follows:

(1) If a field contains tag "/***o***", its field type is "*O*".

(2) If a field contains tag "/***p***", its field type is "*P*".

(3) If a field contains tag "/***r***", its field type is "*R*".

(4) If a field does not contain tags "/***o***", "/***p***", or "/***r***", its field type is "*N*".

(5) If a field contains tags "/***o***" and "/***p***", its field type is "*OP*".

(6) If a field contains tags "/***o***" and "/***r***", its field type is "*OR*".

(7) If a field contains tags "/***p***" and "/***r***", its field type is "*PR*".

(8) If a field contains tags "/***o***", "/***p***", and "/***r***", its field type is "*OPR*".

For example, Table 3 shows three instance candidates in a structural file segment and their position relation schemas. Each instance candidate has four fields, which are all transformed into the corresponding field types of the position relation schemas.

As shown in Table 3, the first position relation schema correctly reflects the position relation implied by the instance candidate. But the second one lacks some organization name, and the third one introduces a distorted "*R*" (contained in "*OR*"), which makes it difficult to determine the right person name. We expect the latter two position relation schemas can have the same form as the first one. For this purpose, we introduce a *normal form* for each structural file segment.

The *normal form* of a structural file segment represents the common correct form of all position relation schemas contained in a structural file segment. Suppose we have the normal form of a structural file segment, we are able to achieve the following goals:

(1) Determination of unknown organization names in some position relation schemas.

(2) Elimination of ambiguities in the field types of some position relation schemas.

(3) Removal of instance candidates in a structural file segment, whose structures largely deviate from the normal form (see section 5.2).

Table 3. Examples of position relation schemas

No.	Instance candidate	Position relation schema
1	姚忠良(Yao Zhongliang)/***r*** 一 男(Male) 一 白象(White Elephant) 食品(Foods) 集团(Group)/***o*** 一 董事长(Board Chairman)/***p***	*<R, N, O, P>*
2	杨宁(Yang Ning)/***r*** 一 男(Male) 一 空中(Kong Zhong) 网(Website) 一 总裁(President)/***p***	*<R, N, N, P>*
3	肖志国(Xiao Zhiguo)/***r*** 一 男(Male) 一 路明(Lu Ming)/***r*** 集团(Group)/***o*** 一 董事长(Board Chairman)/***p***	*<R, N, OR, P>*

In this paper, we present an effective algorithm to determine the normal form (as shown in Figure 4). The algorithm is based on the observation that person name, organization name, and position name of a position relation are not arbitrarily distributed in an instance candidate. In contrast, there are some rules concerning their relative positions in instance candidates. Hence, we classify instance candidates into two types.

(1) Double-Typed Instance Candidates: In these instance candidates, a position relation is implied by exactly two fields, so their position relation schemas contains exactly one field type "*OP*" and another "*R*" and all the remaining field types are "*N*". For example, "大贺集团董事长(the board chairman of Da He Group)—贺超兵(He Chaobin)" is a double-typed instance candidate, whose position relation schema is <*OP*, *R*>. The term "*double-typed*" means there are only two relevant field types appearing in an instance candidate, thus only three field types are allowed to appear in the normal form of double-typed instance candidates, which are "*OP*", "*R*", and "*N*".

(2) Triple-Typed Instance Candidates: A triple-typed instance candidate contains one organization name field, one position name field, and one person name field. Hence, the normal form of those instance candidates should contain exactly one field type "*O*", "*P*", and "*R*", respectively. For example, the first instance candidate in Table 3 is a triple-typed instance candidate, whose position relation schema is <*R*, *N*, *O*, *P*>. The normal forms of triple-typed instance candidates may contain and can only contain field types from the set {*O*, *P*, *R*, *N*}.

So, now our task is to first classify the instance candidates and then develop algorithms to eliminate ambiguities in their fields. For instance, if we have determined that the instance candidate is double-typed with the position relation schema <*OP*, *PR*>, and the normal form is <*OP*, *R*>, we can infer that the field typed with "*PR*" should be a person name according to the normal form. We also know that only field types from the set {*O*, *P*, *R*, *OP, N*} may appear in the normal form.

The detailed algorithm to determine the normal form of instance candidates is shown in Figure 4. We first generate the position relation schemas for all instance candidates, and then we count the frequency of each field type of the relevant set {*O*, *P*, *R*, *OP*}; this information is captured using a matrix *M*[4 × *mfc*]. Finally, we determine the type of the instance candidates and each element of the normal form.

Algorithm *Generating_Normal_Form*

Input: *a structural file segment S*

Output: *F: the normal form of S*

Precondition: *mfc is the number of field types in normal form*

```
1   E ← all the position relation schemas of S;
2   F ← <'N', ..., 'N'>; //total mfc elements with field type 'N'
3   For i ← 1 to mfc Do //Constructing matrix M[4 × mfc]
4       M[1, i] ← count of "O" in the i-th position in E;
5       M[2, i] ← count of "P" in the i-th position in E;
6       M[3, i] ← count of "R" in the i-th position in E;
7       M[4, i] ← count of "OP" in the i-th position in E;
8   End For
9   v1 ← max(M[1, i], i = 1, 2, ..., mfc); suppose v1 = M[1, p1]
10  v2 ← max(M[2, i], i = 1, 2, ..., mfc); suppose v4= M[2, p2]
11  v3 ← max(M[3, i], i = 1, 2, ..., mfc); suppose v4= M[3, p3]
12  v4 ← max(M[4, i], i = 1, 2, ..., mfc); suppose v4= M[4, p4]
13  If v4 > v1 Then //double-typed
14      F[p4] ← 'OP';
15      F[p3] ← 'R';
16  Else //triple-typed
17      F[p1] ← 'O'; F[p2] ← 'P'; F[p3] ← 'R';
18  End If
19  Return F;
```

Figure 4. An algorithm to generate the normal form of a structural file segment

Figure 5 shows an example of the algorithm. Figure 5(a) is the position relation schemas we obtained from the given instance candidates. The matrix describing the frequency of each field type's appearing in different positions in position relation schemas is shown in Figure 5(b). Because the max count of "*OP*" is larger than the max count of "*O*", we conclude that the normal form is double-typed. Since "*R*" appears mostly in the first position, we generate the final normal form as <*R*, *N*, *OP*>.

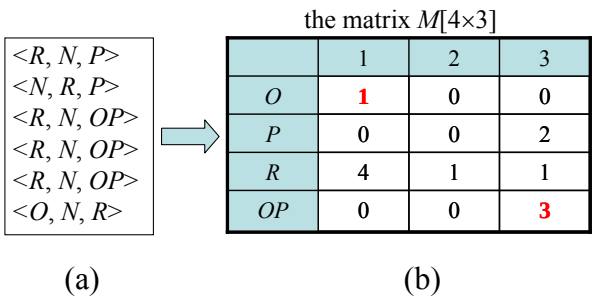

Figure 5. An example of the algorithm Generating_Normal_Form

5.2. Extracting Position Relation Elements

To extract position relation elements from the instance candidates in a structural file segment, we first examine whether the position relation schema of an instance candidate matches the normal form of the structural file segment. If yes, we extract elements from the proper positions in the instance candidate according to the normal form, otherwise we ignore the instance candidate, because it has too many ambiguities.

The crucial problem of extracting position relation elements is to check whether or not the position relation schema of a given instance candidate matches the normal form. Generally, let *F* be the normal form and *S* be a position relation schema, there are four matching strategies between them:

(1) *Left-first*: The matching process proceeds from the left-most positions to the right-most positions of *F* and *S*.

(2) *Organization-name-firs*t: We first find *S*'s matching element with the organization name element in *F*, i.e., "*O*"(triple-typed) or "*OP*"(double-typed), before we start to match the other elements. However, this strategy will always fail if *S* does not contain a field type containing "*O*", which will decrease the performance of our algorithm.

(3) *Person-name-first*: We first find *S*'s matching element with the person name element in *F*, i.e. "*R*", before we start to match the other elements. This strategy is not effective, because it is very likely that more than one person name exists in *S,* due to the fact that many people use a person name to name their organizations, e.g., the organization "张小泉集团 (Zhang Xiaoquan Group)" includes the person name "张小泉 (Zhang Xiaoquan)".

(4) *Position-name-first*: We first find *S*'s matching element with the position name element in *F*, i.e. "*P*"(triple-typed) or "*OP*"(double-typed), before we start to match the other elements. This is the most effective way, because each position relation schemas has a position name. Moreover, a position name is usually unique in *S*, which will further enhance the effectiveness of our matching algorithm.

Hence, we use the *position-name-first* strategy to check whether the position relation schema of a given instance candidate matches the normal form. In our algorithm, we only consider element matches containing "*P*" or "*R*" and neglect the field type "*O*", because a position relation schema will not always have the field type "*O*" according to the instance candidate definition. So, we first remove the field type "*O*" from the normal form, i.e., "*O*"→"*N*" for triple-typed normal forms, or "*OP*"→"*P*"

for double-typed normal forms. The detailed matching algorithm between a position relation schema and its normal form is shown in Figure 6.

Algorithm *Matching*

Input: *a normal form F, containing the elements F*[1], ..., *F*[*mfc*]
a position relation schema E: E[1], ..., *E*[*k*], *k* ≤ *mfc*

Output: *Return true is F matched E, otherwise false*

Precondition: *size*(*F*) = *mfc, i.e. F contains mfc elements. size*(*E*) = *k, k* ≤ *mfc.*

```
1    For t ← 1 to mfc Do //replace 'O' in normal form
2        If F[t] = 'O' Then F[t] ← 'N';
3        If F[t]= 'OP' Then F[t] ← 'P';
4    End For
5    i ←  the index of 'P' in F; //F[i] = 'P'
6    j ←  the index of the element containing 'P' in E;
7    f ← i; e ← j;
8    While (f ≥ 1 and e ≥ 1) Do  //check the left side
9        If F[f] is not contained in E[e]
10           Return false;
11       End If;
12       f––; e––;
13   End While;
14   f ← i; e ← j;
15   While (f ≤ size(F) and e ≤ size(E)) Do //check the right side
16       If F[f] is not contained in E[e]
17           Return false;
18       End If;
19       f++; e ++;
20   End While;
21   Return true;
```

Figure 6. Matching algorithm for a position relation schema and its normal form

Figure 7 shows an example how the matching algorithm works. It first replaces "*OP*" in the normal form with "*P*", then performs the alignment of position names, i.e., it aligns "*P*" in the normal form with "*OP*" in the position relation schema. Finally, it checks the corresponding pairs and returns *true* or *false* as the result.

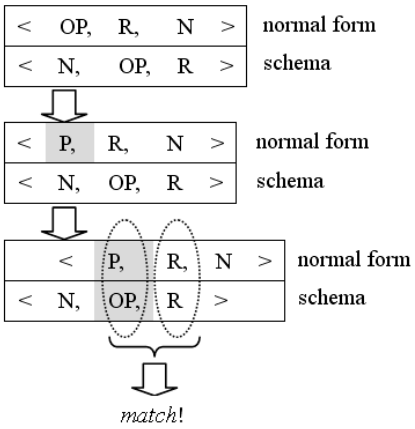

Figure 7. Example of the Matching process

Normal Form	< *OP*, *R*>	
Example 1	*Instance Candidate*	千(Qian) 橡(Xiang) 集团(Group)/***o*** 总裁(president)/***p*** – 陈一舟(Chen Yizhou)/***r***
	Schema	< ***OP***, ***R***>
	Position Relation	< 千 橡 集 团 (QiaXiang Group), 总 裁 (president), 陈一舟(Chen Yizhou)>
Example 2	*Instance Candidate*	盛大(Sheng Da) 总裁(president)/***p*** – 谭群钊(Tan Qiongzhao)/***r***
	Schema	< ***P***, ***R***>
	Position Relation	<盛大(Sheng Da), 总裁(president), 谭群钊(Tan Qiongzhao)>

Figure 8. Examples of extracting position relation elements

After we have identified the instance candidates whose position relation schemas match the normal form of the structural file segment containing all instance candidates, we can easily extract position relation elements from each matching instance candidate. Figure 8 shows two examples. The position relation consists of three elements, namely an organization name, a position name, and a person name. In the first example, we obtain person name "陈一舟(Chen Yizhou)" and a field "千(Qian) 橡(Xiang) 集团(group)/*o* 总裁(president)/*p*" using the *Matching* method shown in Figure 7. Next, we can directly extract the organization name and the position name from the field having type "*OP*" identified by the tags "/***o***" and "/***p***". However, in the second example, we are only able to extract a field having type "*P*" in the matching step. In our current approach, we scan the fields backwards and get the position name by using a dictionary of position names. In this example, the position name scanned is "总裁(president)", and then the remainder of the field, "盛大(Sheng Da)", is regarded as the organization name.

6 Performance Evaluation

6.1. Settings and Dataset

To conduct our experiment, we downloaded 6028 Web pages from the famous Chinese search engine *Baidu*. Our keywords used look like "position name + Chinese surname" , such as "总裁(president)+张(Zhang)丨王(Wang)丨李(Li)丨赵(Zhao)丨刘(Liu)". This method increases the probability that a Web page contains position relation instances. For our experiment, we choose five kinds of position relations (president, manager, engineer, CEO, board chairman). The position name dictionary contains 66 simplified Chinese position names which are prepared manually.

We evaluate the experimental results of our approach based on recall *R*, precision *P,* and F-measure *F*. F-measure considers both recall and precision; hence, it can be considered as a trade-off measure. The formula of F-measure is:

$$F-measure=\frac{(1+\beta^2)P*R}{\beta^2*P+R} \tag{1}$$

β is a trade-off coefficient. If β=1, recall is assessed as equally important as precision. If β is less than 1, recall is weighted less than precision. In our experiment, β is assigned 0.5, denoting that precision is more important than recall.

In our experiment, the number of position relations that should be recalled is set to the amount of position relations existing in the structure file segments when the structural coefficient threshold is set to 0 and the max field count is 7.

We performed four experiments to make a thorough performance study of our approach: (1) person-name complement experiment, (2) position-relation extraction experiment, (3) impact of the structural coefficient on the overall performance and (4) impact of the max field count on the performance.

The test computer is equipped with a Pentium (R) 4 CPU with 3.0 GHz, 512 MB main memory, and is running Windows XP Professional with Service Pack 3. All the algorithms are implemented using C# language, a programming tool in Visual Studio 2005.

6.2. Experimental Results

6.2.1. Person Name Complement

ICTCLAS, which we use for person name tagging in Section 4.3, fails to recognize given names in some sentences. These sentences will be eventually filtered out when running the generating algorithm for instance candidates, due to the lack of person name tags in the sentences. We introduce the person-name complement policy (see Section 4.3) to solve this problem. The following experiment is to demonstrate its effectiveness.

The experiment is conducted on 18501 sentences with surname tag "/*nr1*". These sentences are selected from the sentences extracted from 6028 Web pages using the method described in Section 4.2 and applying the condition that the sentence must contain surname tag "/*nr1*". We randomly choose

1000 sentences as a test example and use recall and precision to evaluate our method. If a surname in a sentence is complemented to form a complete name - (see Section 4.3), the resultant sentence is considered as a recalled sentence.

Among the 1000 sentences, 713 sentences are tagged with right surnames by ICTCLAS, whereas the remaining 287 sentences have wrong surname tags. Each sentence consists of one person name tag. We perform our experiment on the 713 sentences and try to complement person names based on the rules discussed in Section 4.3.

In our experiment, we obtained the following results:

(1) Among the 713 surname tags generated by ICTCLAS, 438 person names are complemented by our algorithm, where 427 have correct person names, i.e., the precision is about 97%.

(2) We also collect the complemented person names contained in the position relations extracted. Among the final 28512 position relations returned, 3184 contain the complemented person names. To this extent, our algorithm for person name complement contributes about 11% to the final result of position relations extracted.

6.2.2. Position Relations Extraction

Table 4. Experimental results for five position relations

	Quantity recalled	**Recall**	**Precision**	**F-measure**
总裁 (president)	4991	88.2%	97.0%	95.1%
经理 (manager)	12490	84.3%	95.1%	92.7%
工程师 (engineer)	2547	88.1%	90.1%	89.7%
董事长 (board chairman)	9027	90.1%	98.7%	96.9%
首席执行官 (CEO)	613	84.8%	95.4%	93.1%
Sum/Average	29668	87.2%	96.1%	94.2%

Table 4 shows the extraction results of five kinds of position relations over the 1425 structural file segments obtained when the structural file segment threshold is 0.9 and MFC is 7. The average recall is over 87%, whereas the precision of our approach is much higher. The reason why our approach gains such a high precision is due to the use of structural features of position relations in Web pages. Once these structural features are correctly described, the position relations will be extracted accurately. Our approach uses a normal form to represent these features in an appropriate way and adopts a matching method using alignment to to check the applicability of instance candidates when extracting position relations. Both policies improve the precision of our approach. The reason of obtaining relatively low recall is that structural segments in some Web pages are not as regular as tables or lists. Instance candidates in such structural segments often have different fields, which increase the failure rate of our matching algorithm. Finally, F-measures of different kinds of position

relations considerably vary from each other. This is caused by the differing representation styles of those position relations in Web documents.

6.2.3. Impact of the Structural Coefficient

The structural coefficient of a file segment reflects its structural level. A high structural coefficient is derived if only few ambiguities occur in the file segment. A high structural coefficient threshold may cause high precision but low recall, while a low structural coefficient threshold might denote high recall but low precision. To get satisfactory results for both precision and recall, a proper structural coefficient is crucial in our approach.

Table 5. Impact of the structural coefficient α

α	Recall	Precision	F-measure	Quantity of newly recalled pos. relations	Precision of newly recalled pos. relations
1	0.253	0.973	0.620	8498	0.973
0.99	0.847	0.965	0.938	20220	0.961
0.98	0.859	0.963	0.940	463	0.88
0.97	0.864	0.962	0.941	206	0.83
0.96	0.866	0.962	0.941	43	0.907
0.95	0.868	0.962	0.941	87	0.736
0.925	0.870	0.961	0.941	104	0.846
0.9	0.872	0.961	0.942	47	0.894
0.8	0.872	0.961	0.942	40	0.7
0.7	0.873	0.961	0.942	11	0.82
0.6	0.873	0.961	0.942	3	1
0.3	0.873	0.960	0.942	20	0.75
0	0.873	0.960	0.942	0	—

In order to measure the influence of the structural coefficient on the overall performance, we change the structural coefficient threshold (α) from 0 to 1 and observe precision, recall, and *F*-measure. Moreover, the quantity and precision of newly recalled position relations are also computed. The result is shown in Table 5. The recall gained by the experiment is only 0.253 when the structural coefficient threshold equals 1, denoting that there are only few complete structural file segments (having no distortions). But when the structural coefficient is slightly reduced, the recall rapidly increases, whereas the precision is only a little reduced, which indicates that most position relations exist in structural file segments with few distortions. As soon as the structural coefficient threshold is less than 0.99, it has little influence on recall, precision, and F-measure, especially when it is reduced from 0.9 to 0. This is due to the number of newly recalled position relations, which is quite small

compared to all recalled position relations, meaning that the structural file segments with a low structural coefficient have only few position relations. Moreover, we notice that the precision of newly recalled position relations declines at the same time. This shows that the structural file segments with a low structural coefficient have more distortions and tend to cause more errors for the position relation extraction. Based on the analysis above, the proper structural coefficient threshold should be between 0.9 and 0.99.

6.2.4. Impact of the Max Field Count (MFC)

The max field count (*MFC*) is introduced to limit the number of fields an instance candidate can have. In some cases, position relation instances in Web pages are mingled with other information such as gender (see Table 3), age, and so on. Such instances often contain more than 3 fields and will be filtered out if *MFC* is 3. Thus, the value of *MFC* should at least be 3. To recall the instance candidates consisting of more than 3 fields, we should set *MFC* to a value greater than 3. However, on the other side, a high *MFC* value will introduce more complexity into the extracting process of position relations.

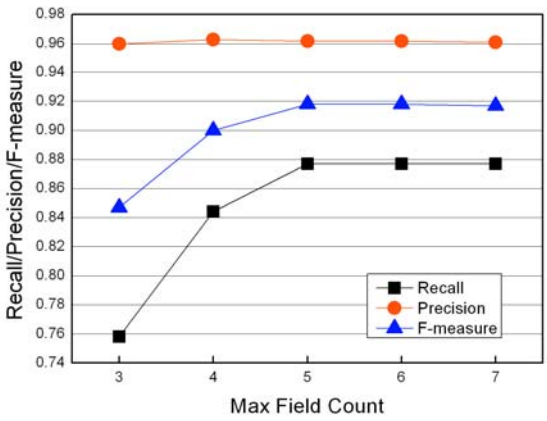

Figure 9. Influence of the max field count (MFC)

Figure 9 shows the influence of *MFC* on recall, precision, and F-measure. Here, the structural coefficient is set to 0.9, and we use the same dataset as we did in the previous experiment. As Figure 9 shows, a proper *MFC* value over 5 is acceptable.

7 Conclusions and Future Work

In this paper, we made an investigation concerning the structural feature of Web contents and proposed a structure-based framework to extract position relations from the Web. We conducted experiments on a set of Chinese Web pages, and the experimental results confirm that our approach is effective to extract different types of position relations.

Based on the discussion in this paper, we draw the following conclusions:

(1) Many Web pages contain structural contents, which is very useful for the extraction of competitive intelligence. While we only focus on the structure-based extraction of position relations in

this paper, it may also be useful for the extraction of other intelligence information, such as competitors' products, services, investors, co-operators, and so on.

(2) Concentration on the structural parts of Web pages is helpful to extract position relations with high precision. Our experiment shows that precision is generally over 96% when the structure-based extraction method is used.

(3) Our current approach is restricted to the Chinese language. But the most important contribution of the paper is the framework to utilize the structural feature of Web pages to extract information for competitive intelligence from the Web. For English Web pages, we need to replace the fundamental tool for person name extraction and revise the approach to generate instance candidates.

In our future work, we will study the feasibility of our structure-based extraction method for English Web pages. Furthermore, we will focus on business relation extraction based on the structural feature of Web pages.

Considerable work still remains to be carried out in this area, both in terms of further validation of the underlying hypothesis, and in more detailed exploration of the proposed process. In particular, it would be important to understand how developers (and analysts in particular) can distil requirements from prototypes and the associated client feedback. This involves a clearer theoretical framework for understanding client uncertainty and how various prototypes reduce this uncertainty.

Acknowledgements

We are grateful to Prof. Theo Härder and Prof. Jiangliang Xu for their constructive advice on improving the paper. This work is supported by the National Science Foundation of China under the grant no. 71273010 and no. 60776801, the National Science Foundation of Anhui Province (no. 1208085MG117), and the USTC Youth Innovation Foundation.

References

1. Agichtein, E., & Gravano, L., Snowball: Extracting Relations from Large Plain-text Collections. In Proceedings of the Fifth ACM International Conference on Digital Libraries, 2000, 85-94
2. Brin, S., Extracting Patterns and Relations from the World-Wide Web. In Proceedings of the 1998 International Workshop on the Web and Databases (WebDB'98), 1998, 172-183
3. Kim, S., Jeong, M., Lee, G. G., Ko, K., & Lee, Z., An Alignment-based Approach to Semi-supervised Relation Extraction Including Multiple Arguments. In Proceedings of AIRS, LNCS 4993, 2008, 526-536
4. Li, W. G., Liu, T., & Li, S. Automated Entity Relation Tuple Extraction Using Web Mining. ACTA Electronica Sinica, 2007, 35(11): 2111-2116
5. Ravichandran, D., & Hovy, E. Learning Surface Text Patterns for a Question Answering System. In Proceedings of the ACL Conference, 2002, 41-47
6. Reichartz, F., Korte, H. and Paass, G., Dependency Tree Kernels for Relation Extraction from Natural Language Text. In Proceedings of ECML/PKDD, 2009, 270-285
7. Giuliano, C., Lavelli, A., Pighin, D., & Romano, L., FBK-IRST: Kernel Methods for Semantic Relation Extraction. In Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), 2007, pp.141-144
8. Huang, R., Sun, L., & Feng, Y., Study of Kernel-Based Methods for Chinese Relation Extraction, In Proceedings of AIRS, LNCS 4993, 2008, pp.598-604

9. Zelenko, D., Aone, C., & Richardella, A., Kernel Methods for Relation Extraction. Journal of Machine Learning Research, 2003, 3: 1059-1082
10. Zhao, S. B., & Grishman, R., Extracting Relations with Integrated Information Using Kernel Methods. In Proceedings of the 43rd Annual Meeting of the ACL, 2005, pp.419-426
11. Zhang, Y., Xu, X., & Zhang, T., Fusion of Multiple Features for Chinese Named Entity Recognition Based on CRF Model, In Proceedings of AIRS, LNCS 4993, 2008, pp.95-106
12. Yao, L., Sun, C., Wang, X., & Wang, X., (2010) Combining Self Learning and Active Learning for Chinese Named Entity Recognition, Journal of Software, 2011, 5(5): 530-537
13. Liu, Y., Jin, P., Yue, L., Extracting Position Relations from the Web, In Proceedings of 11th ACM International Workshop on Web Information and Data Management (WIDM'09), Hong Kong, China, 2009, pp. 59-62
14. ICTCLAS, http://www.ictclas.org (2008, accessed April 2012)
15. Jin, P., Chen, H., Lin, S., Zhao, X., Li, X., & Yue, L., Indexing Temporal Information for Web Pages, Computer Science and Information Systems, 2011, 8(3): 711-737
16. Jin, P., Li, X., Chen, H., Yue, L., CT-Rank: A Time-aware Ranking Algorithm for Web Search, Journal of Convergence Information Technology, 2010, 5(6): 99-111