Web Information Extraction for the Creation of Metadata in Semantic Web

Ching-Long Yeh and Yu-Chih Su
Department of Computer Science and Engineering
Tatung University
Taipei, Taiwan
chingye@cse.ttu.edu.tw g9206033@ms2.ttu.edu.tw

Abstract

In this paper, we develop an automatic metadata creation system using the information extraction technology for the Semantic Web. The information extraction system consists of preparation part that takes written text as the input and produces the POS tags for the words in the sentences. Then we employ finite state machine technology to extract the units from the tagged sequences, including complex words, basic phrases and domain events. We use the components of an NLP software architecture, GATE, as the processing engine and support all required language resources for the engine. We have carried out an experiment on Chinese financial news. It shows promising precision rate while it need further investigation on the recall part. We describe the implementation of storing the extracted result in RDF to an RDF server and show the service interface for accessing the content.

1. Introduction

Semantic Web is an emerging technology working by building a metadata layer upon the current web and using the metadata description language to describe the resources on the WWW [1]. The metadata layer is a structured information layer, that is, the information model of RDF [2]. The service programs built upon the metadata layer can provide an efficient way for users to find the information they need according to the concepts. Furthermore, the metadata layer supports agents to provide the function of service automation for users, called Semantic Web Service [3]. The semantic metadata layer can be constructed either in a manual way using annotation tools, such as Annotea [4] or automatically using the natural language processing technology [5]. The former approach is good at higher precision rate while suffered in efficiency. The latter approach can be used to sort out the problem of the other; however, it needs very high cost to achieve similar result using the manual way.

Information extraction is a low-cost approach to natural language processing using the finite-state automata technology to extract specific noun sets and information matching specific syntax and semantic templates [6]. In this paper we employ the finite-state automata technology to extract information from the resources on the WWW to serve as the describing information of the resources, that is, the metadata. The extracted metadata is then used as a part in the Semantic Web.

Ontology, the semantic schema used to build metadata layer upon semantic web, contains the concept hierarchy of specific domain knowledge and the detail features of all concepts type. In this paper we build the templates for information extraction based on the feature structure of concepts type of ontology.

A typical information extraction system, such as FASTUS of SRI [7], ANNIE of the University of

Sheffield [8], is constructed by using a cascade of finite-state grammars. Based on such mechanism, we can identify the units of words, complex words, basic phrase group, co-reference resolution, identifying event structures *etc.*, in text. In our implementation, we use components of GATE framework [8] to develop our Chinese information extraction system. The system in sequence consists of the following modules: word segmentation, part-of-speech (POS) tagging, name entity extraction, noun group extraction and event extraction. The word segmentation module is based on matching entries in the lexicon with the input sentences. We develop a Brill POS tagger [3] in order to provide accurate POS information for latter modules. Then the succeeding extraction modules are developed by using the finite-state machine technology. The JAPE (Java Annotation Processing Engine) engine is used as the basis of the extraction modules. The task of developing the extraction functions is therefore to define the regular expressions of the respective extraction objects.

The output of the information extraction system is converted into RDF and is imported into an RDF indexing system, Sesame [9]. Therefore user can access the content of the extracted metadata through the conceptual search services interfaces provided by Sesame.

In brief, our goal is to build an ontology-driven information extraction system that processes sentences and transforms extracted data into RDF automatically. The extracted metadata is used to build a knowledge base. Services of conceptual search and semantic navigation are implemented based on the inference engine of the knowledge base. In Section 2, we describe the architecture of the Semantic Web and the related representation languages used to construct the metadata layer. In Section 3, we describe the architecture of the information extraction and integration with the Semantic Web architecture. In Section 4 we describe components of the information extraction system. In Section 5, we describe the implementation and the performance of the system.

2. Architecture and Representations of Semantic Web

In this section, we first describe the metadata layer of the Semantic Web and system architecture for managing system the layer. Then we describe the representations of the metadata layer.

2.1 Metadata Layer of Semantic Web

Semantic Web is an extension of the current Web where information is given well-defined meaning, better enabling computers and people to process in cooperation. Resources on the Web are given explicit meaning using markup language to make up a metadata layer in addition to the current information pool as summarized in Fig. 1. The Web Ontology Language (OWL) [10] provides rules for defining knowledge structures, i.e., ontology, in order that instances of knowledge can be created based on the common structures. OWL is an extension to the Resource Description Framework (RDF) [11] by adding vocabularies used to define the knowledge structures instead of the tree structures in XML documents.

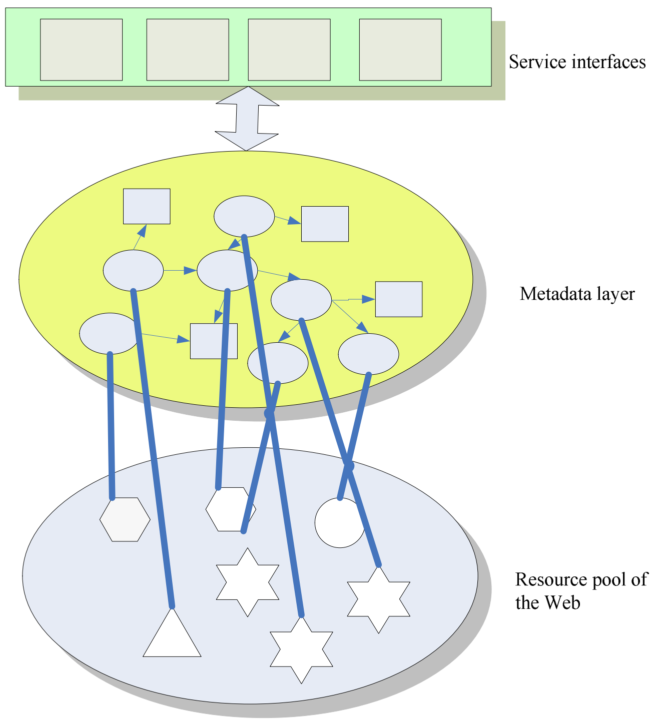


Figure 1: Architecture of the metadata layer of the Semantic Web

The Semantic Web framework can be summarized as providing a metadata layer in content-interoperable languages, mainly in RDF, which intelligent or automatic services can be made by machines based on the layer. The typical architecture for managing the metadata layer, for example, KA2 [12] and Sesame [13], consists of an ontology-based knowledge warehouse and inference engine as the knowledge-based system to provide intelligent services at the front end, such as conceptual search and semantic naviagtion for user to access the content as summarized in Fig. 2. In the backend is the content provision component, consisting of various tools for creating metatdata from unstructured, semi-structured and structured documents. In this paper, the information extraction system we developed is used as a component of the back end.

2.2 Representing the Schema and Instances of the Metadata Layer

The metadata layer of the Semantic Web as shown in Figure 1 is built on RDF. The concptual model of RDF is a labled directed graph [11], where the nodes in the graph are identified by using URIs [14] or literals representing resources and atomic values of some kinds, respectively. The label on the directed arc connecting two nodes, from a recource to another, or a resource to a literal, represents the relationship between both ends. An arc, also identified using URI, connecting two nodes is a triple of the subject-predicate-object, similar to the structure of simple declarative sentetence. Thus an RDF document can be seen as a list of triples. A triple states a fact about a subject, a resource on the Web. Since the nodes in RDF are identified using the addressing scheme of the Web, an RDF document can be easily combined with other to form an integrated description of resources on the Web. For example, the graph shown in Figure 3 represents “The author of http://www.cse.ttu.edu.tw/chingyeh is Ching-Long Yeh”.

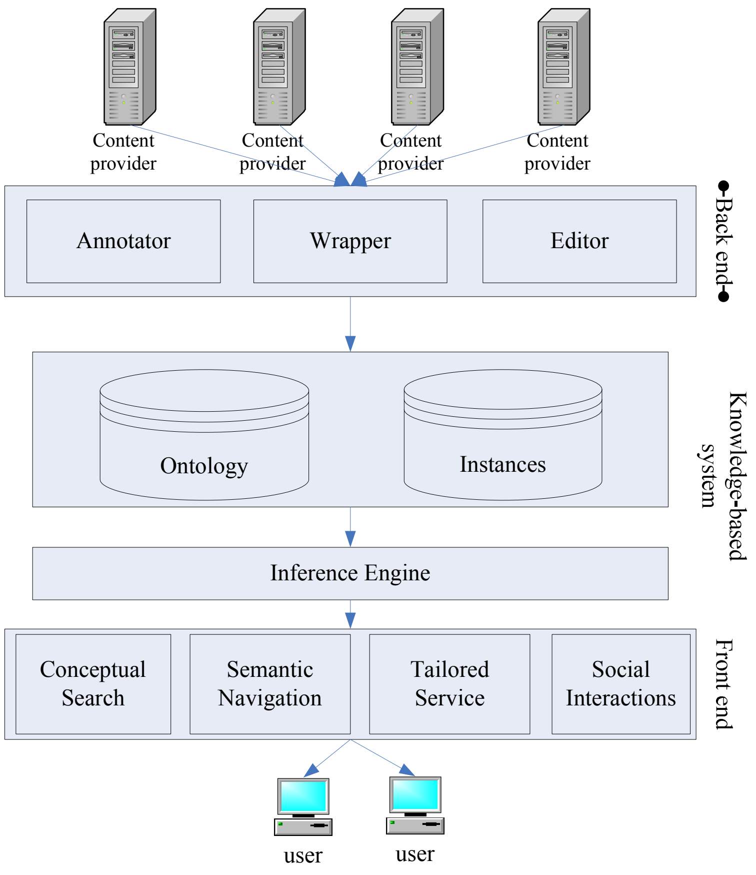


Figure 2: System architecture for managing the metadata layer of Semantic Web

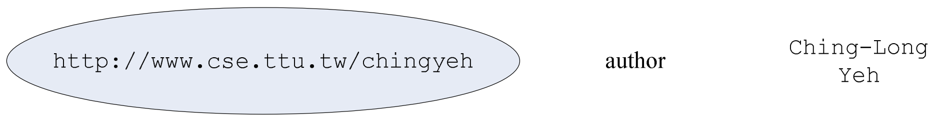


Figure 3: An RDF fragment

Representing in XML it becomes as follows.

```
<rdf:RDF
    xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
    xmlns:t='http://www.cse.ttu.edu.tw/sw'>
   <rdf:Description about="http://www.cse.ttu.edu.tw/chingyeh">
     <t:author>Ching-Long Yeh</t:author>
    </rdf:Description>
</rdf:RDF>
```

The information layer represented by using RDF is a generic relational data model, describing the relationship between resources or between resource and atomic values. The meaning of the resources can then be found in the domain knowledge base, that is, ontology. The representation of ontology in the Semantic Web is an extension of RDF, OWL [10].

3. Information Extraction System and Metadata Creation of the Semantic Web

As mentioned previously, information extraction can provide the function of creating metadata for resources on the web. Thus we integrate the information extraction function as a part of the back end of the Semantic Web system as described in Section 2. The system we design consists of three parts: information extraction back end, ontology-based store and service front end as shown in Figure 4. The work of the back end is to extract the domain events from the relevant documents returned by search engine. The ontology-based store converts the extracted data to specific formats and stores them into the repository. We have implemented two kinds of stores; one is RDF store and the other is a frame-based store. In this paper, we focus on the former one. Finally, the service front end provides interface for user to access the extracted content.

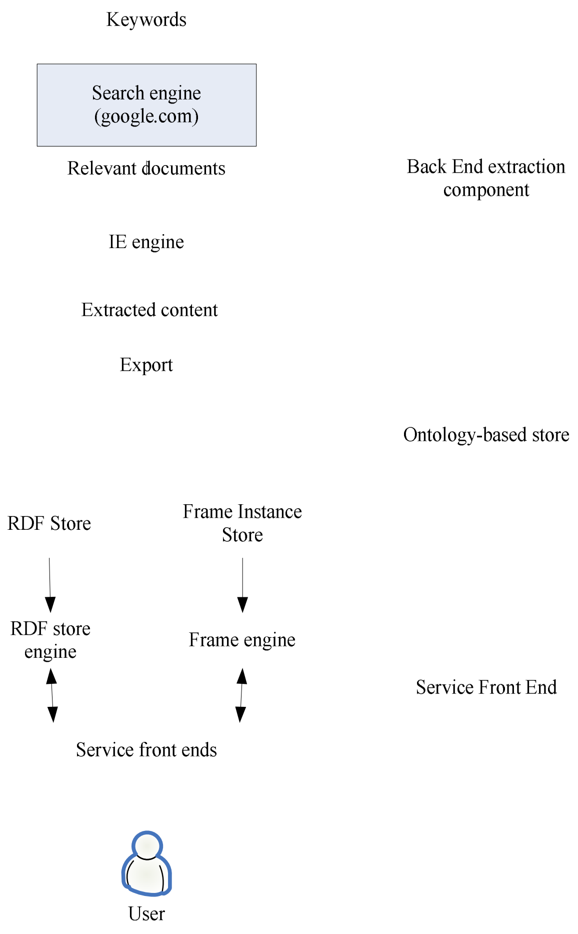


Figure 4: Integrating information extraction function with Semantic Web system

The back end of the system includes several information extraction components, including Chinese Tokenizer, Gazetteer, Sentence Splitter, POS tagger and Name-entity Transducer, as shown in Figure 5. They are used to extract specific domain events from a large number of relevant documents in a cascaded way. The relevant documents we use here are returned from calling a search engine and served as the domain data. The flow of domain events extraction starts from Chinese Tokenizer which is used to recognize the basic tokens, that is words. Gazetteer is used to recognize the special terms of domain, for example, number, day, etc. Sentence splitter and POS tagger are used to tag each token with its correct part-of-speech. Finally, the Name-entity transducer is the real work used to recognize the domain events. All processing flow is shown in the following figure.

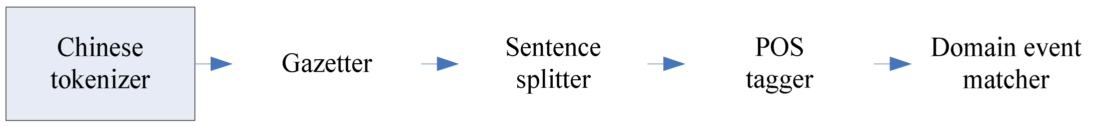


Figure 5: The components in the information extraction system.

The central part of the system, the ontology-based store, consists of two parts: transformation mechanism and repository. The transformation mechanism transforms the original extracted data into different specific format according to the interest of users. It works as if a common interface which takes extracted domain events as input and exports the specific format data as output. After exporting the specific format data, we should have a place to store the data so that our query service has a knowledge source to query. So we should provide specific repository for specific data format to store.

The front end of the system consists of several services, including conceptual search, explore service, extraction service and clear service. The conceptual search is a search which attempts to conceptually match results with the query. It does not match by the key words, rather their concepts. The function of this service is to help user to find the data. The explore service is to explore the data according the classes and properties of ontology. The extraction service is used to extract all data from the repository. The clear is used to delete all data from the selected repository. The goal of front end is to provide several services based on semantic web for users to use.

4. Information Extraction System

Instead of building up such framework from scratch, here we adopt the reusable software component provided by the GATE framework [8] to construct the information extraction system for our purpose. Actually this is the approach having been used by ANNIE [8], a Nearly-New Information Extraction System. It consists of components used in the task of information extraction; furthermore, it provides a pipelined environment to chain all relevant components together. After all required components are added, the pipeline executes these components in a cascaded way and the results returned from current or previous components are kept so that the succeeding component can use these results to do further process. The information extraction system built by using the components in GATE provides the necessary engines; we therefore focus our attention on developing the knowledge resources necessary for each component in the system.

The information extraction system consists of five components, which can be divided into two parts, the preparation and the extraction phases as shown in Fig. 5. The preparation part is composed of the front four components, that is used to process the raw input sentences and produces word sequence tagged with their correct part-of-speech information. The other part is used to extract domain events. In the following, we describe these two parts in order.

4.1 Preparation phase

We describe in order the components in the preparation phases as shown in Fig. 5.

Chinese Tokenizer

Chinese Tokenizer is used to segment the input sentences into sequences of meaningful units, namely tokens, and identify their types such as words, numbers and punctuation. The knowledge sources of the tokenizer include a dictionary and ambiguity resolution rules. For the former part we employ the Chinese dictionary of the CKIP [15]. For the latter part we only employ the “long-term first” heuristic rule for disambiguation.

Once a word is matched, it is assigned with a type. For example, the three tokens “籃球”, “1010” and “，” are assigned the type “words”, “numbers” and “punctuation”. For the unmatched case from the current character by looking up the dictionary, we assign the type “words” to that character and then continue the matching work from the next character.

Gazetteer

The Gazetteer is used to define various categories of specific domain in which each category contains a large number of instances with the same concept. It is very convenient for later domain events processing. By grouping the instances of the same concept in advance, in the course of domain events recognition, we can just easily use the concept rather than the redundant words to construct the patterns. By reusing the function of ANNIE Gazetteer, we just need to focus on the definition of domain key words. For the Gazetteer function to function properly, it needs one or more `.lst` files and a `.def` file.

A `.lst` file is simply a file containing the key words of specific concept. For instance `company.lst` contains the names of company in every line of the file. The file `lists.def` is used to describe which `.lst` files we use. Except for the name of `.lst` file, we have to define the major type which represents more general concept of that file and, optionally, a minor type which represent more specific concept of that file. Here we just use the major type for our purpose. For each `.lst` file, we define the major type served as their individual concept, so we can later use the concept rather than the large number of words to recognize the domain events. This is very useful feature in the course of domain events recognition.

```
Company.lst:Company
day.lst:day
date_key.lst:date_key
hour.lst:hour
time_ampm.lst:time_ampm
production.lst:production
…
```

Sentence Splitter

The purpose of sentence splitter is to segment the full text into sentences. This component is required for the POS tagger because the processing of POS tagger is in the way of sentence by sentence. The design of sentence splitter is easy. We just need to define the symbols representing the end of a sentence. Then, segment the full text into sentences according these symbols. In this paper we the sentential mark , “。”, comma, “，” and question mark “？” as the symbols of the end of sentences.

POS tagger

The POS tagger is used to tag each word recognized by the Chinese Tokenizer with its part-of-speech. By tagging each word with the part-of-speech in advance, in the course of domain events recognition, we can just use the POS rather than the redundant words to construct the patterns.

There are two steps to tag each word with part-of-speech. In order to label each word with its most-likely part-of-speech, we first need a lexicon dictionary in which each word is tagged with its most-likely part-of-speech. Then, use the initial state processor to label each word recognized by the Chinese Tokenizer with its most-likely tags by looking up the lexicon dictionary. We employ the Chinese balanced corpus developed by the CKIP [16] in this step.

Second, use the contextual rules to filter out the incorrect part-of-speech according to the contextual information. We can not find existing POS tagger to be employed in this paper; we therefore develop a POS tagger by employing Brill algorithm [5].

4.2 Domain events extraction

After the basic words and their features obtained in the preparation phase, the extraction phase further processes to obtain information about domain events. We employ the method used in FASTUS [7] to process the domain event from processing simpler units, the complex words to phrasal constituents, basic phrases, and finally domain events. All information produced by this stage is recognized by using the JAPE rules; we describe how to formulate JAPE rules at each processing step.

Complex words

The purpose of processing complex words is to identify the string of company, location, production, date, money, and year *etc*. First, we define JAPE rule by using the features produced by the gazetteer as the condition, for example, the rule of company word as below.

```
Rule: company
(
 {Lookup.majorType == Company}
):com
-->
 :com.Company = {kind = "company"}
```

It is obviously not sufficient to identify the complex words by simply using the features returned by the gazetteer. In Chinese, a noun phrase has the head-final feature [17]. In other words, the head noun appears in the final position of a noun phrase. Here we treat the feature word returned by the gazetteer as the head noun, and its preceding adjacent nouns as its modifiers. This observation is formalized as a number of JAPE rules for complex words. For example, if one or more noun connects together with the company identifier, then we recognize the preceding nouns as a company word. The company identifier is defined in the stage of Gazetteer, including “公司”, ”集團”, “航空” and so on.

```
Macro: COMPANY_IDENTIFIER
({Lookup.majorType == company identifier})
Rule: companyWithIdentifier
(
 ({Token.category == Na})+
  (COMPANY_IDENTIFIER)
):com
-->
 :com.Company = {kind = "company"}
```

In the following is another example of complex word rule for the recognition of money.

```
Rule: money1
//Priority:30
(
 (NUMBER)
 ({Token.category == PERIODCATEGORY}|{Token.string == "."})?
 (NUMBER)?
 ({Token.category == Neu})
 {Token.category == Nf}
 (MONEY_KIND)?
):money
-->
 :money.Money = {kind = "money"}
```

The recognized complex words are then used for later processing of basic phrases and domain events.

Basic phrases

The purpose of processing basic phrases is to identify several word classes, including noun group, verb, preposition, conjunction etc. For simple classes such as verb, preposition, conjunction etc, we can just use the part-of-speech tagged to each word by POS tagger to recognize them. The example using the POS to recognize the simple classes is as follows.

```
Rule: preposition
//Priority:30
(
 {Token.category == P}
):preposition
-->
 :preposition.Preposition = {kind =
"preposition"}
```

For the complicated classes like noun group, we combine determiner and other modifiers together with the head noun to form a noun phrase. For example, in the following example, the part-of-speech “Neu” and “Nf” together with the noun, we can recognize the noun group like “三本書”or “四間分公司”.

```
Rule: nounGroup1
//Priority:30
(
 {Token.category == Neu}
 {Token.category == Nf}
 {Token.category == Na}
):noungroup
-->
 :noungroup.NounGroup = {kind = "NounGroup"}
```

Domain events

In order to recognize the domain events, we first have to survey in which events we are interested from the relevant documents according to domain knowledge. After finding the event, we then define the pattern based on the form of that event. For example, we are interested in the news about which company is merged by which company. So, we look for this kind of event from the relevant documents. Once we find this kind of event, such as “明基與西門子在 7 日下午宣佈合併”, we can define the pattern like the following format.

“{Company}{Conjunction}{Company}…{Date}?...{合併}”

All keyword encompassed by the brackets are already prepared in the previous processing, so we can easily use this pattern to define our JAPE rule as following. Here we take advantage of the gazetteer to define the relevant words in advance, including “合併”, “併購” and “收購”. With this convenient way, we do not use a large number relevant word (“合併”,”併購”,”收購”) rather than the concept (MERGE) to recognize the domain event.

```
Macro:MERGE
({Lookup.majorType == merge})
Rule: pattern1
//Priority:30
(
 {Company.kind == company}
 {Conjunction.kind == conjunction}
 {Company.kind == company}
 ({Token.kind == word})*
 ({Date.kind == date})?
 ({Token.kind == word})*
 (MERGE)
):goal
-->
 :goal.Pattern = {kind = "pattern"}
```

5. Implementation

In this section we first show the implementation of domain events extraction according to the design described previously. Then, we investigate the performance of an experiment we conducted. Finally, we describe the implementation of storing the extracted result in RDF store and the services provided by the store.

5.1 Information Extraction System

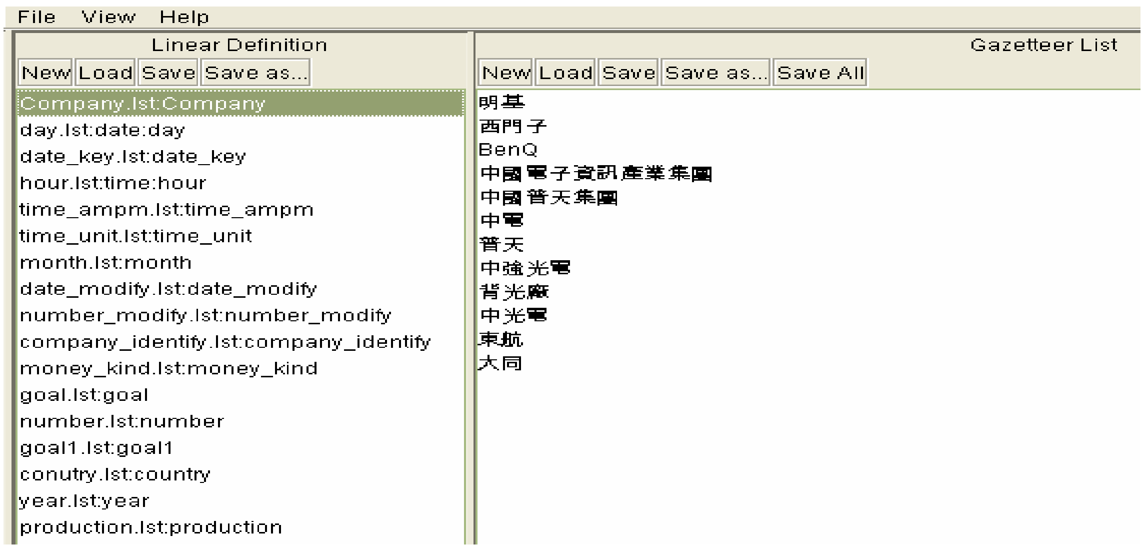
The GATE framework provides a pipelined environment in which each component is executed in a sequent way [8]. By using GATE, what we need to do first of all is to select the required components, as described in Section 4, from GATE and place then in proper order. This becomes the engine of the information extraction system. We have described the function of each component in Section 4. Here we focus on the description of the knowledge source that we provide for each component.

Chinese Tokenizer

The knowledge source for the tokenizer is a dictionary. The dictionary we employ is based on the Chinese Electronic Dictionary developed by the CKIP group [15]. Since it is in XML format, we employ the DOM (Document Object Model) parser [18] to retrieve the Chinese tokens within the dictionary and store in the format required by the tokenizer. The number of Chinese tokens we use in the tokenizer is about 140000.

Gazetteer

Based on the function of GATE Gazetteer, we can just focus on the definition of domain keywords. We list the financial domain categories used in our domain events extraction as shown in Fig. 6. The left panel lists all the categories we define and the right hand side shows the list of keyword of the selected category.

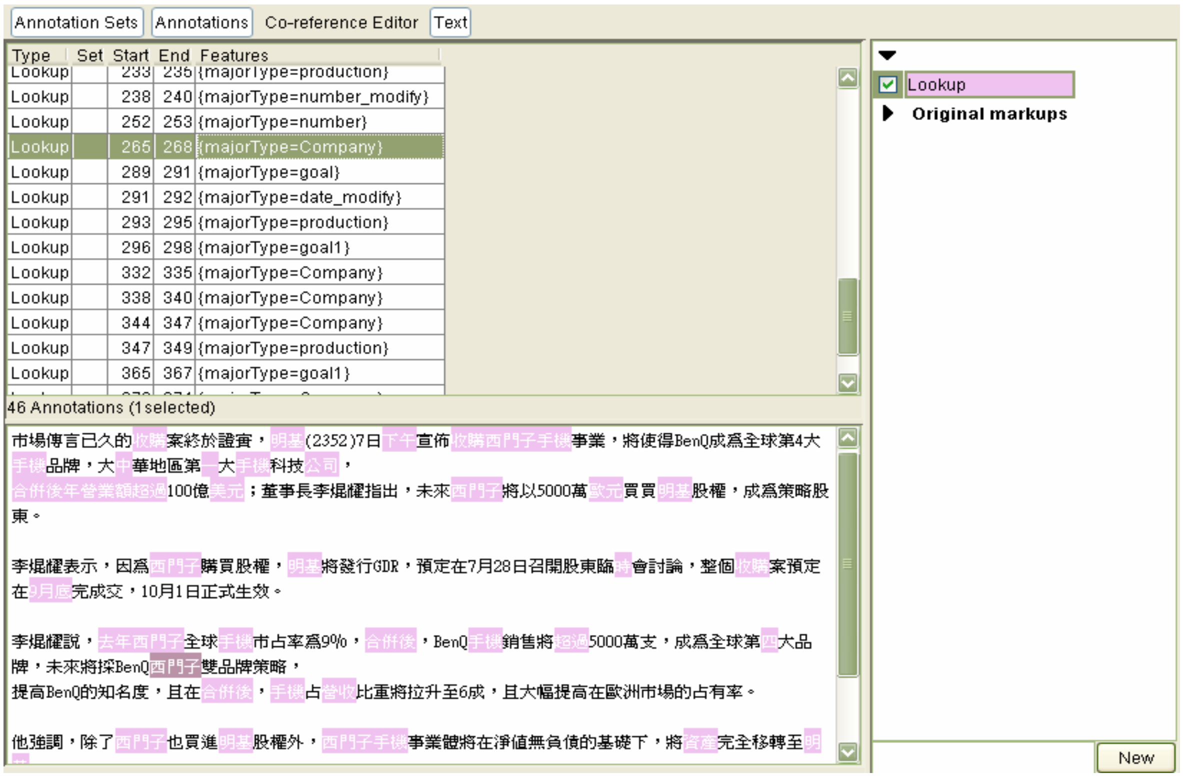


Company.lst:Company
day.lst:date:day
date_key.lst:date_key
hour.lst:time:hour
time_ampm.lst:time_ampm
time_unit.lst:time_unit
month.lst:month
date_modify.lst:date_modify
number_modify.lst:number_modify
company_identify.lst:company_identify
money_kind.lst:money_kind
goal.lst:goal
number.lst:number
goal1.lst:goal1
conutry.lst:country
year.lst:year
production.lst:production

明基
西門子
BenQ
中國電子資訊產業集團
中國普天集團
中電
普天
中強光電
背光廠
中光電
東航
大同

Figure 6: Definitions of the categories of domain keywords

In Fig. 7, we show the result as the document passes through the Gazetteer. We can see that several domain key words defined in our categories are found from the document. And, once a key word is found, it is attached with a `majorType` as its feature.



Type	Set	Start	End	Features
Lookup		233	236	{majorType=production}
Lookup		238	240	{majorType=number_modify}
Lookup		252	253	{majorType=number}
Lookup		265	268	{majorType=Company}
Lookup		289	291	{majorType=goal}
Lookup		291	292	{majorType=date_modify}
Lookup		293	295	{majorType=production}
Lookup		296	298	{majorType=goal1}
Lookup		332	335	{majorType=Company}
Lookup		338	340	{majorType=Company}
Lookup		344	347	{majorType=Company}
Lookup		347	349	{majorType=production}
Lookup		365	367	{majorType=goal1}

46 Annotations (1 selected)

市場傳言已久的收購案終於證實，明基(2352)7日下午宣佈收購西門子手機事業，將使得BenQ成爲全球第4大手機品牌，大中華地區第一大手機科技公司，
合併後年營業額超過100億美元；董事長李焜耀指出，未來西門子將以5000萬歐元買買明基股權，成爲策略股東。

李焜耀表示，因爲西門子購買股權，明基將發行GDR，預定在7月28日召開股東臨時會討論，整個收購案預定在9月底完成交，10月1日正式生效。

李焜耀說，去年西門子全球手機市占率爲9%，合併後，BenQ手機銷售將超過5000萬支，成爲全球第四大品牌，未來將採BenQ西門子雙品牌策略，
提高BenQ的知名度，且在合併後，手機占營收比重將拉升至6成，且大幅提高在歐洲市場的占有率。

他強調，除了西門子也買進明基股權外，西門子手機事業體將在淨值無負債的基礎下，將資產完全移轉至明

Figure7: Result produced by the Gazetteer

Sentence Splitter

Here we use the sentential mark , “。”, comma, “，” and question mark “?” in Chinese text to segment the text into sentences.

POS tagger

The GATE framework provides a Brill POS tagger [19]. In this paper we develop the rule base for the tagger by using the training algorithm of the Brill tagger [5]. We employ the Chinese balanced corpus developed by the CKIP [16] for the training and testing purposes. In our implementation, when training contextual rules on 550000 words with threshold 10, an accuracy of 95.43% is achieved. When we reduce the number of words to 55000 words, the accuracy is dropped to 94.46%. Finally, when we use only 16000 words to train the contextual rules, the accuracy is just 93.48%. The comparison is as follows.

Number of words	threshold	Accuracy (%)	Number of errors	Rule count
550000 words	10	95.43%	25469	341
55000 words	10	94.46%	3033	43
16000 words	10	93.48%	1063	8

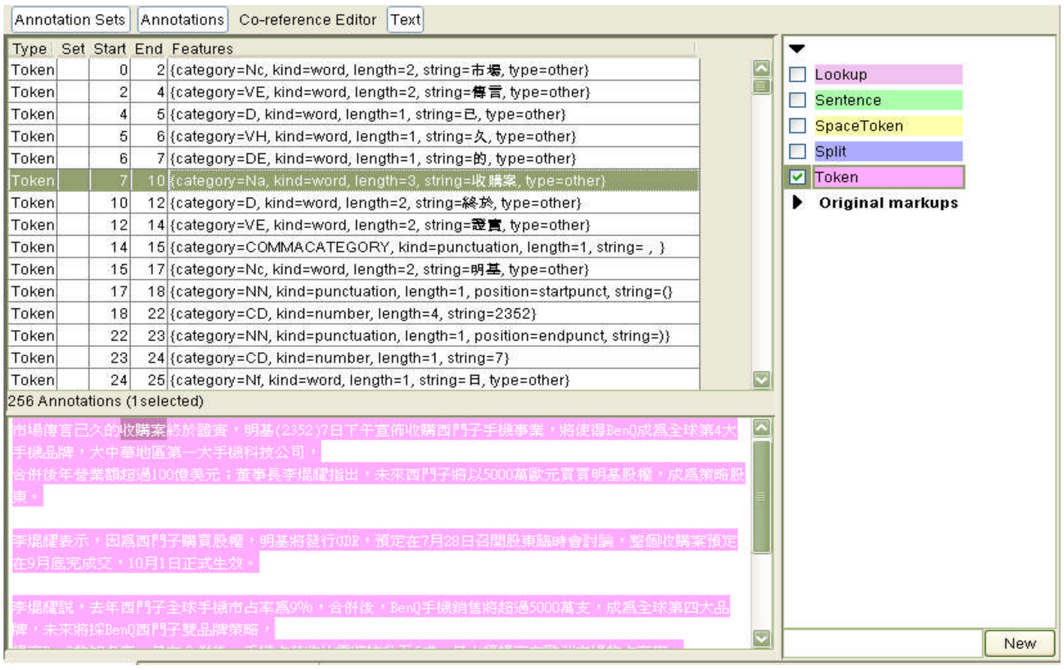
Based on this experiment, we figure out the size of the training data has the direct impact on the accuracy. The more training data the learner use, the more contextual rules can be obtained, so, the more errors can be corrected. Next we apply these rules to the test data and see how many errors can be corrected. The test data we used here is 20000 words. The result is as follows.

Number of words	threshold	Accuracy (%)	Number of errors	Rule count
Initial state		93.7%	1280	No rule
550000 words	10	95.6%	894	341
55000 words	10	94.41%	1136	43
16000 words	10	93.76%	1268	8

Another reason which is related to the accuracy is the threshold. In the following experiment, we find that the less the threshold, the more precise the accuracy.

Brill Tagger	threshold	Accuracy (%)	Error number	Rule count
		93.34%	36797	No rule
	17	95.14%	27111	215
	14	95.30%	26254	274
	12	95.35%	25925	302
	10	95.43%	25469	341

The result of using the POS tagger is shown in Fig.8.



Type	Set	Start	End	Features
Token		0	2	{category=Nc, kind=word, length=2, string=市場, type=other}
Token		2	4	{category=VE, kind=word, length=2, string=傳言, type=other}
Token		4	5	{category=D, kind=word, length=1, string=已, type=other}
Token		5	6	{category=VH, kind=word, length=1, string=久, type=other}
Token		6	7	{category=DE, kind=word, length=1, string=的, type=other}
Token		7	10	{category=Na, kind=word, length=3, string=收購案, type=other}
Token		10	12	{category=D, kind=word, length=2, string=終於, type=other}
Token		12	14	{category=VE, kind=word, length=2, string=證實, type=other}
Token		14	15	{category=COMMACATEGORY, kind=punctuation, length=1, string=，}
Token		15	17	{category=Nc, kind=word, length=2, string=明基, type=other}
Token		17	18	{category=NN, kind=punctuation, length=1, position=startpunct, string=(}
Token		18	22	{category=CD, kind=number, length=4, string=2352}
Token		22	23	{category=NN, kind=punctuation, length=1, position=endpunct, string=)}
Token		23	24	{category=CD, kind=number, length=1, string=7}
Token		24	25	{category=Nf, kind=word, length=1, string=日, type=other}

256 Annotations (1selected)

市場傳言已久的收購案終於證實，明基(2352)7日下午宣佈收購西門子手機事業，將使得BenQ成爲全球第4大手機品牌，大中華地區第一大手機科技公司，
合併後年營業額超過100億美元；董事長李焜耀指出，未來西門子將以5000萬歐元買買明基股權，成爲策略股東。

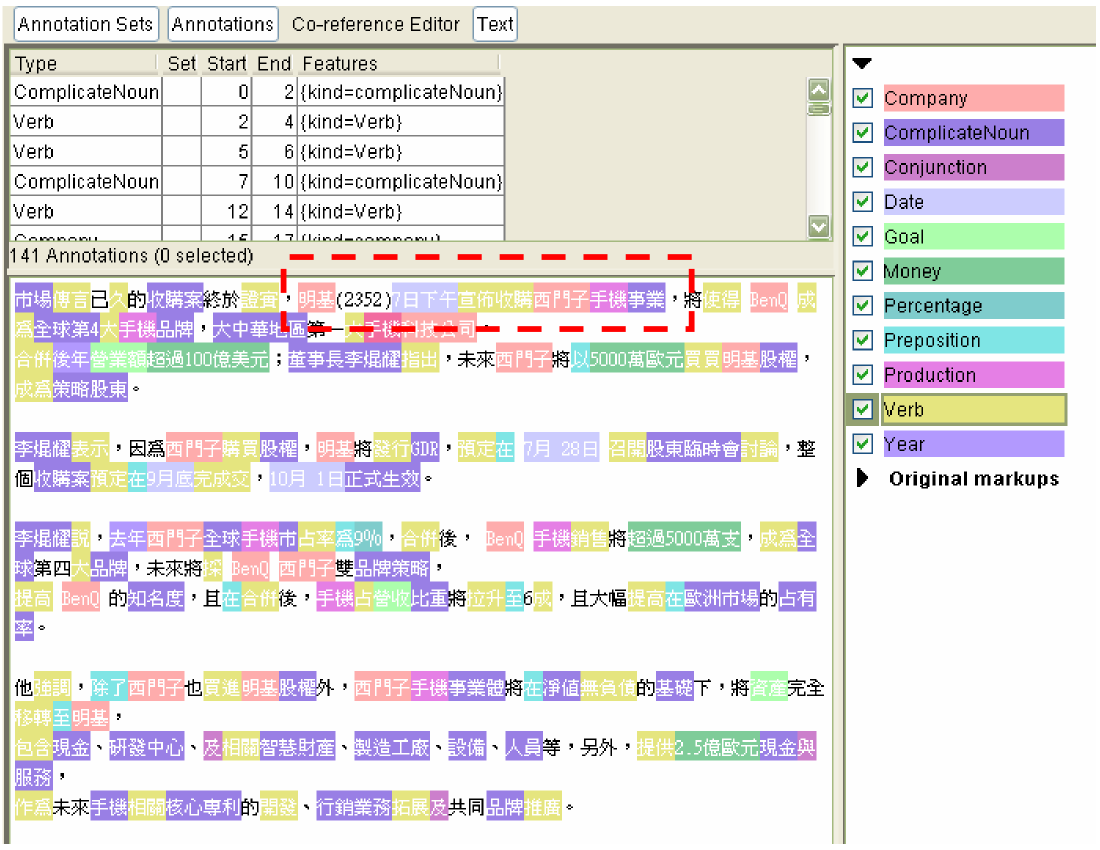
李焜耀表示，因爲西門子購買股權，明基將發行GDR，預定在7月28日召開股東臨時會討論，整個收購案預定在9月底完成交，10月1日正式生效。

李焜耀說，去年西門子全球手機市占率爲9%，合併後，BenQ手機銷售將超過5000萬支，成爲全球第四大品牌，未來將採BenQ西門子雙品牌策略，

Figure: 8 Result of POS tagger

Domain events extraction

As described in Section 4, we can build up JAPE rules to extract the complex words, basic phrases and domain events from the output produced by the preceding components. The main task here is on the definition of JAPE rules. As they have been described in Section 4, here we only show the result of the implementation. Since the complex words and basic phrases are used to contribute the extraction of domain events. We therefore skip the smaller units while show the result of domain event. For example, the pattern "{Company}{Date}{收購}{Company}{Product}?{Detail}?" can be used to extract the event "明基(2352)7 日下午宣佈收購西門子手機事業" as shown in Fig. 9.



Type	Set	Start	End	Features
ComplicateNoun		0	2	{kind=complicateNoun}
Verb		2	4	{kind=Verb}
Verb		5	6	{kind=Verb}
ComplicateNoun		7	10	{kind=complicateNoun}
Verb		12	14	{kind=Verb}
Company		15	17	{kind=company}

141 Annotations (0 selected)

市場傳言已久的收購案終於證實，明基(2352)7日下午宣佈收購西門子手機事業，將使得 BenQ 成爲全球第4大手機品牌，大中華地區第一大手機科技公司。
合併後年營業額超過100億美元；董事長李焜耀指出，未來西門子將以5000萬歐元買買明基股權，成爲策略股東。

李焜耀表示，因爲西門子購買股權，明基將發行GDR，預定在 7月 28日 召開股東臨時會討論，整個收購案預定在9月底完成交，10月 1日正式生效。

李焜耀說，去年西門子全球手機市占率爲9%，合併後， BenQ 手機銷售將超過5000萬支，成爲全球第四大品牌，未來將採 BenQ 西門子雙品牌策略，
提高 BenQ 的知名度，且在合併後，手機占營收比重將拉升至6成，且大幅提高在歐洲市場的占有率。

他強調，除了西門子也買進明基股權外，西門子手機事業體將在淨值無負債的基礎下，將資產完全移轉至明基，
包含現金、研發中心、及相關智慧財產、製造工廠、設備、人員等，另外，提供2.5億歐元現金與服務，
作爲未來手機相關核心專利的開發、行銷業務拓展及共同品牌推廣。

✓	Company
✓	ComplicateNoun
✓	Conjunction
✓	Date
✓	Goal
✓	Money
✓	Percentage
✓	Preposition
✓	Production
✓	Verb
✓	Year

Figure 9: Result of extracting domain events

5.2 Evaluation

We first take one hundred financial news returned from the search engine, google, as our test data and use dozens of JAPE rules to recognize the specific domain event, after the processing of domain events matching, we then calculate the precision rate and recall rate of our system. We first manually extract the target events within these financial news and we obtain 120 events of interest. After the domain event matching, it returns 25 results, among which 22 are correct. So the precision rate is 88%, and the recall rate is 18%. The precision rate shows a promising result of using the information extraction system; however, the recall rate we obtain obviously needs further investigation as follows.

- First of all, the scope of JAPE rule recognition we use here is between the symbols, such as "，", "。", "?", *etc*. That is, the JAPE rule would not be able to extract events that cross the boundary of sentence symbol. for example, "中國東方航空以九億八千六百萬元人民幣，向母公司收購西北航空公司的業務". To solve this problem, we can divide our domain events recognition into several parts. For example, we can first process the recognition of one sentence, after all one sentence is processing, and we then process the recognition of two sentences, and so on.
- Second, using Part-of-speech identifier to recognize the domain events may result in error recognition. When we use the part-of-speech to recognize the domain event, some words which are irrelevant but belong to that part-of-speech may be recognized as the domain event. This will produce error domain events.
- Finally, the JAPE rules we use here is not sufficient to cover all the situations. So, if we

encounter some specific domain event that our JAPE rules do not define, we will lose it. To solve this problem, we further need much investigation into the expansion of the JAPE rules in order to improve the recall rate.

5.3 RDF store and the Services

The RDF store is used to store the RDF we obtained from the result information extraction system. In this paper, we employ an RDF store from the open source, Sesame [9]. Sesame is an ontology-based RDF store. It accepts RDF files as input and stores the input files in its indexing system. In the front end, it provides a number of application program interfaces for service programs to access the content of the RDF contents. In brief, Sesame play the functions of the ontology-based store and the front end service as shown in Fig. 4 in Section 3. The extracted domain contents are converted into RDF format and then the result can be imported into Sesame using its API. After the extracted contents are stored in the RDF store of Sesame, we then can use the query and explore service interfaces to access the content in Sesame.

Sesame provides SeRQL as its query language. For detailed description, please refer to the user guide of Sesame [20]. We give a brief description here. For example, if we want to query an RDF graph for persons who work for companies that are IT companies, the query for this information can be in the RDF graph in Fig. 10.

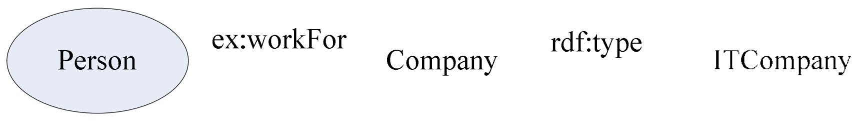


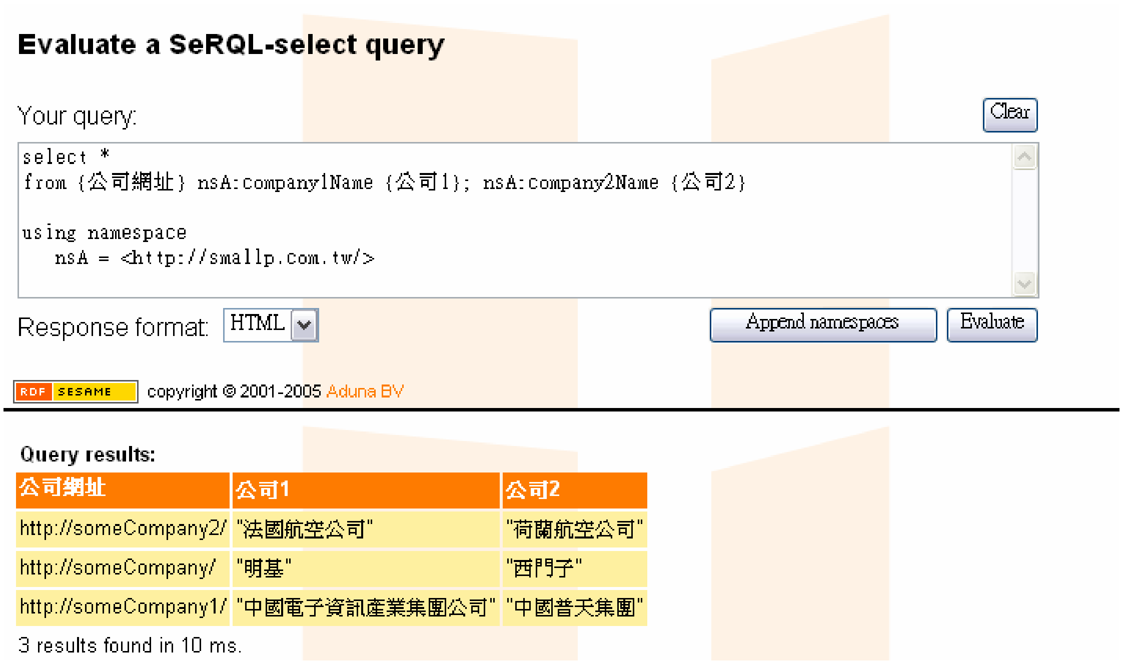
Figure 10: Example of query graph for SeRQL

The SeRQL path notation for the above RDF is written as below.

```
{Person} ex:worksFor {Company} rdf:type {ITCompany}
```

The parts enclosed by brackets represent the nodes in the RDF graph, the parts between these nodes represent the arcs in the graph. The SeRQL query language supports two query concepts. The first type of queries are called “select query”, the second type of queries are called “construct query”. Here we just use the select query because the form of select query is like the form of SQL, it is easy for us to use it to construct the conceptual search interface. The select clause determines what is done with the results that are found. It can specify which variable values should be returned and in what order. The from clause is optional and always contains path expressions. It defines the paths in an RDF graph that are relevant to the query. The where clause is optional and can contain additional constraints on the values represented in the path expression.

With the select query, we can extract whatever information we want to know. For example, in our financial domain, we may want to know which company is merged by which company. So we can write a simple query sentence as following to extract this information.



Evaluate a SeRQL-select query

Your query:

```
select *
from {公司網址} nsA:company1Name {公司1}; nsA:company2Name {公司2}

using namespace
   nsA = <http://smallp.com.tw/>
```

Response format: HTML

copyright © 2001-2005 Aduna BV

Query results:

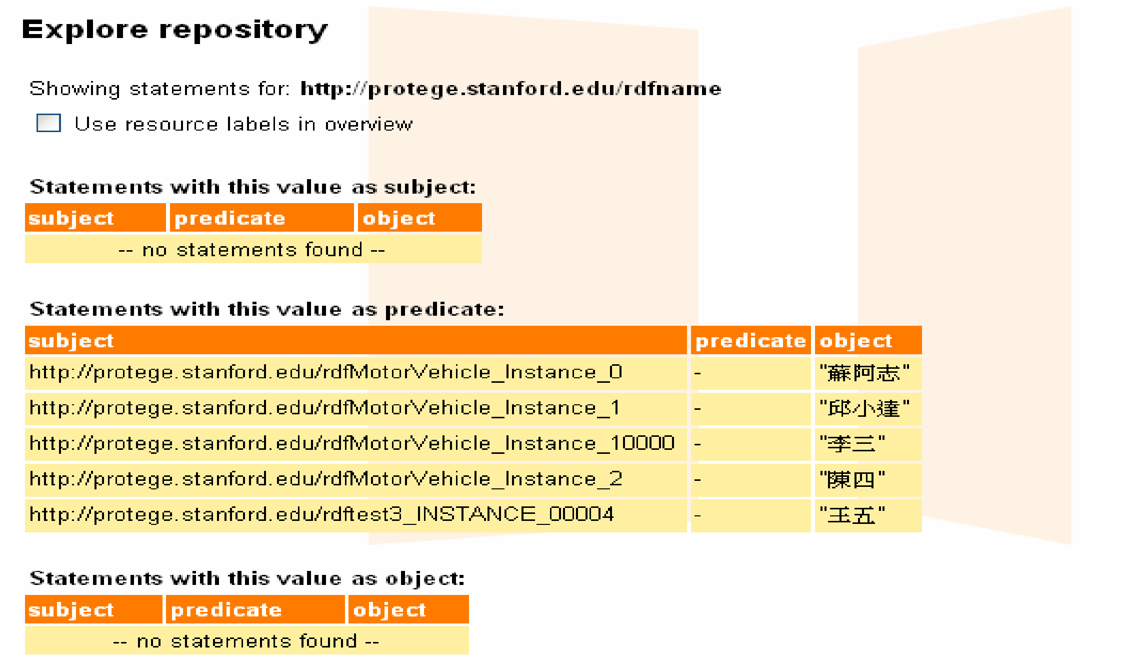
公司網址	公司1	公司2
http://someCompany2/	"法國航空公司"	"荷蘭航空公司"
http://someCompany/	"明基"	"西門子"
http://someCompany1/	"中國電子資訊產業集團公司"	"中國普天集團"

3 results found in 10 ms.

Figure 10: Example of using SeRQL of Sesame

Explore Service

The explore service can do the exploration according to the concept, that is, Class or Property. There are two methods to do the exploration. This first method is to enter the URI ourselves according to which concept you want to explore. The second method is to use the information provided by the Sesame. We can just click the URI represented in the following, and it will explore that URI for us. As shown in Fig. 11 is the result when we want to explore the property `http://protege.stanford.edu/rdfname`. We can find that when this property serves as a predicate, we can get all names of people existing in our RDF data.



Explore repository

Showing statements for: **http://protege.stanford.edu/rdfname**

☐ Use resource labels in overview

Statements with this value as subject:

subject	predicate	object
-- no statements found --		

Statements with this value as predicate:

subject	predicate	object
http://protege.stanford.edu/rdfMotorVehicle_Instance_0	-	"蘇阿志"
http://protege.stanford.edu/rdfMotorVehicle_Instance_1	-	"邱小達"
http://protege.stanford.edu/rdfMotorVehicle_Instance_10000	-	"李三"
http://protege.stanford.edu/rdfMotorVehicle_Instance_2	-	"陳四"
http://protege.stanford.edu/rdftest3_INSTANCE_00004	-	"王五"

Statements with this value as object:

subject	predicate	object
-- no statements found --		

Figure 11: Example of exploring the content in Sesame

6. Conclusions

We have carried out automatic creation of metadata for the Semantic Web using the information extraction technology. The extracted information is stored in an RDF server and can be accessed through the service interfaces of the RDF server. In this paper we successfully use the components of the GATE software architecture for NLP as the processing engine to build up the system. Thus we focus our attention on the language knowledge sources required for the components. The reuse of the software components really shortens the creation of the prototype of the system. Since the system is developed mainly using the finite state machine technology, we obtain excellent system performance. We have tested the system on Chinese financial news. The result shows that most of the extracted domain events can be correctly identified. However, we still need do further investigation obtain wider coverage of extraction. In particular, we need to pay more attention on the creation of extraction rules for domain events.

Acknowledgements

We would to give our thanks to CKIP, Academia Sinica for their support of the balanced corpus.

References

[1] W3C, Semantic Web, http://www.w3.org/2001/sw/

[2] W3C, RDF Primer, W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-primer/

[3] David Martin, *et al.*, Bringing Semantics to Web Services: The OWL-S Approach, *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, 2004, San Diego, California, USA.

[4] Jose Kahan, Marja-Riitta Koivunen, Eric Prud'Hommeaux, and Ralph R. Swick, Annotea: An Open RDF Infrastructure for Shared Web Annotations**,** in *Proc. of the WWW10 International Conference*, Hong Kong, May 2001.

[5] Daniel Jurafsky and James H. Martin, *Speech and Language Processing*, Prentice Hall, 2000.

[6] Douglas E. Appelt and David Israel, Introduction to Information Extraction Technology, IJCAI-99 Tutorial, 1999, Stockholm, Sweden.

[7] Jerry R. Hobbs and David Israel, FASTUS: An Information Extraction System, http://www.ai.sri.com/~israel/fastus_brief2.pdf

[8] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*. Philadelphia, July 2002.

[9] Sesame, http://www.openrdf.org/

[10]W3C, OWL: Web Ontology Language Reference. W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-ref/

[11]W3C. RDF/XML Syntax Specification (Revised). W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-syntax-grammar/

[12] R. Benjamins, D. Fensel, S. Decker, and Gomez-Perez. KA2: building ontologies for the Internet:

a mid term report. *International Journal of Human Computer Studies*. pp. 687-712. 1999.

[13] J. Broekstra, A. Kampman and F. van Harmelen. Sesame: a generic architecture for storing and querying RDF and RDF Schem. In J. Davies, D. Fensel and F. van Harmelen (eds.) *Toward the Semantic Web*: *Ontology-based Knowledge Management*. Wiley. 2003.

[14] T. Berners-Lee, R. Fielding and L. Masinter, Uniform Resource Identifiers (URI): Generic Syntax, IETF RFC: 2396, 1998, http://www.ietf.org/rfc/rfc2396.txt?number=2396

[15] The CKIP Group, Chinese Electionic Dictionary, http://www.aclclp.org.tw/use_ced_c.php

[16] The CKIP Group, Academia Sinica Balanced Corpus, http://www.aclclp.org.tw/use_asbc.php

[17] Charles N. Li and Sandra A. Thompson, 1981. *Chinese – A Functional Reference Grammar*, University of California Press.

[18] The Apache XML Project, Xerces Java Parser 1.4.4 Release, http://xml.apache.org/xerces-j/

[19] M. Hepple. Independence and commitment: Assumptions for rapid training and execution of rule-based POS taggers. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, Hong Kong, October 2000.

[20] B.V. Aduna, User Guide for Sesame, http://openrdf.org/doc/sesame/users/userguide.html.