國 立 交 通 大 學

管理學院（資訊管理學程）碩士班

碩　士　論　文

應用自然語言處理於自動化資訊擷取--
以資訊產品規格之擷取為例

Apply Natural Language Process in Automatic Information Extraction: The Example of IT Product Specification Extraction system

研究生：彭俊彥

指導教授：楊千 教授

中 華 民 國 九十四 年 六 月

應用自然語言處理於自動化資訊擷取--
以資訊產品規格之擷取為例

Apply Natural Language Process in Automatic Information Extraction: The Example of IT Product Specification Extraction system

研 究 生：彭俊彥　　Student: Chun-Yen Peng
指導教授：揚千 教授　　Advisor: Professor Chyan Yang

國 立 交 通 大 學

管理學院（資訊管理學程）碩士班

碩 士 論 文

A Thesis

Submitted to Institute of Information Management

College of Management

National Chiao Tung University

In Partial Fulfillment of the Requirements

For the Degree of

Master of Business Administration

in

Information Management

January 2005

Hsinchu, Taiwan, the Republic of China

中 華 民 國 九十四 年 六 月

應用自然語言處理於自動化知訊擷取－以資訊產品規格之擷取為例

研 究 生：彭俊彥　　　　　指導教授：揚千 教授

國立交通大學 資訊管理研究所

摘　要

經過十多年的發展，網際網路已成為一個龐大的知資載體，然而大部份的資訊都是以自然語言的形式呈現。以自然語言所表達的資訊，雖然便於人類閱覽，但卻無法讓電腦理解。因此如何從大量的網頁中自動擷取隱含的知識，成為目前知識工程研究的一大課題。

資訊擷取的相關研究主要在於取得非結構化文件中的資訊片段，並轉換為較容易為電腦處理和分析的結構化文件格式，目前大致有 Text Mining 和 Web Mining 兩種主要的方式。前者大多與自然語言處理(National Language Processing, NLP)技術相結合，而後者則是將Data Mining的技術運用於網頁的資訊擷取上。

本研究主要是試著結合自然語言處理技術和本體論(Ontology)的概念，建立一個資訊產品規格擷取的系統，以做為自動化網頁資訊擷取的雛型研究。我們運用自然語言處理工具對HTML文件進行適當的處理，再透過JAPE(Java Annotation Patterns Engine)語法規則來擷取文件中特定語法結構的資訊片段，最後再參考預先定義的Ontology，將擷取到的資訊輸出為符合RDF(Resource Description Framework)規範的檔案。

我們針對HP和IBM的網頁撰寫適當的JAPE語法規則，再從這兩家公司的個人電腦(包含Desktop和Notebook)，Unix伺服器，顯示器及印表機等四大產品線，抽樣36個網頁進行產品規格的資訊擷取，其平均召回率(Recall)和正確率(Precision)都超過90%以上，由此可以驗証JAPE語法規則針對特定領域的資訊擷取有相當不錯的效能表現。

關鍵字：資訊擷取、自然語言處理、本體論

Apply Natural Language Process in Automatic Information Extraction: The Example of IT Product Specification Extraction system

Student: Chun-Yen Peng　　　　　Advisor: Dr. Chyan Yang

Institute of Information Management

National Chiao Tung University

Hsinchu, Taiwan, Republic of China

Abstract

World Wide Web (WWW) is a large repository of information that includes many resources, like text document, image, multimedia and so on. Most of information is presented in natural language and disperse in different Web sites. Natural language documents are for human reading but not computer. How to extract the information form natural language document is an important topic of knowledge engineering.

The major task of information extraction is to extract information piece from unstructured document and convert to structured document for computer processing and analysis. There are two major methodologies for information extraction, one is test mining and another is web mining. The former always link to natural language processing technology whereas the latter applies data mining to process the web contents.

Our research tries to build up a prototype system that combines the natural language process and the concept of ontology to extract IT product specification information from web page. We use NLP tools to process HTML document and extract information entities by JAPE grammar rule, then refer to predefined ontology to convert the extracted information to be DAML file that follow RDF specification.

We develop the optimal JAPE grammar rules for IBM and HP web pages that describe IT product specification and download 36 IT product web pages to test the extraction performance. The testing result show the average recall and precision are over 90%. It reveals the JAPE grammar rules have good extraction performance when optimized for specific knowledge domain.

Keyword: NLP, Information extraction, JAPE, grammar rule,

誌　謝

這篇論文得以順利完成，實應歸功於各位師長及親友的協助和勉勵。首先要感謝的是指導教授揚千博士在研究過程中的指導和啟發，確立整體研究方向和步驟。其次要感謝口試委員李其健博士及劉敦仁教授在口試過程中給予的寶貴的建議，使整體論文更加完整。感謝博士班陳良駒學長和楊耿杰學長在研究過程中提供各項重要的參考資料及行政資源上的協助，讓論文得順利進展。

另外也要感謝服務單位-台灣積體電路公司資訊系統建構處的同仁，在我研修期間所給予工作上的支援與協助，使我能投入更多的時間在究研工作上。由衷地感謝洪莉娜、齊玉仙等友人協助論文最後的校稿和定稿等工作，使論文得順利付梓。

最後，謹以這篇論文獻給我敬愛的父母親和家人，感謝他們在我研修期間所付出的關懷和鼓勵，這是支持我在繁忙工作中依然能堅持到底的最大動力。

彭俊彥　謹致

2005 仲夏於

交通大學資訊管理研究所

Contents

Table List

Figure List

1. Introduction

1.1 Background

The economy age evolve from farm economics to industrial economics. Today, most people agree we are entering the knowledge economy age. Michio Kaku claims that "knowledge and technology" will become the only determining factor in a nation's competitiveness [1]

The concept of knowledge economic changes the management of enterprise and makes knowledge management become one of the main competitiveness of business. For business view, information technology plays a crucial role that enhances the performance of knowledge management, such as knowledge retrieval, knowledge store, especially knowledge sharing.

With the dynamic environment and the knowledge economy coming, knowledge has been treated as one of the most important assets that can enhance competitive advantages. For a company to lead among competitors, it is important to ensure that the best corporate knowledge must be available and applied to the needs of the clients in the right places at the right times [2]. Thus, how to creating and sharing knowledge to keep high competition of enterprise is a critical mission of IT managers.

The knowledge/information extraction and presentation are important process in knowledge management. Information Extraction (IE) is an important approach to automated information management. IE is the task of converting documents containing fragments of structured information embedded in other extraneous material into a structured template or database-like representation [3]. The major concern of IE is how to address specific pieces of data in natural language document and extracting structured information from unstructured text.

World Wide Web (WWW) is a large repository of information. Include many resources, like text document, image, multimedia and so on. All of resources can be retrieved by anyone who connects to Internet. Most of information is presented by unstructured text document in natural language and disperse on different site. Since most of web documents are presented by natural language, it is unreadable for computer to extract knowledge from web page, we need an efficient approach to convert natural language document to be computer readable format.

The traditional way to extract information form web page is through search engine to select related document and annotate these documents by manual, much time and effort is needed for information extraction. How to extract information from Internet efficiently is the major concern of our research.

However, recent advances in natural language processing (NLP) open the new choose to perform document annotation and information extraction task. Through the customized and pre-defined process flow, NLP tools could extract information accurately form web page in specific domain [4].

In other and, the concept of Semantic Web and Ontology provide new thinking model about information presentation. Semantic Web is the representation of data on the World Wide Web. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners.

Traditional Web language focus on web page presentation, most of information content still descript in natural language. Although WWW provide a user friendly and platform independent client for information exchange, it can't meet the requirement for automatic process of software agent. The key point is software agent can't understand the information content of web page.

Building computer readable Web pages is one of the terminal goals of the W3C Semantic Web. The idea is first mentioned by Tim-Berner's Lee at the original proposal of WWW at CERN when 1989. The proposal includes a figure showing how information about a web of relationships amongst named objects could unify a number of information management tasks.[5]

An ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. An ontology defines the common words and concepts used to describe and represent an area of knowledge. This definition is consistent with the usage of ontology as set-of-concept-definitions, but more general. And it is certainly a different sense of the word than its use in philosophy [6].

After more than 10 years developing, WWW accumulate millions web page that contain very large information. How to extract the information and translate to be computer readable data format is a key process for Semantic Web promotion, it is also major concern of our research.

1.2 Motivation and Objectives

For most of company and individual, to collect product specification for comparison and evaluation is necessary before IT product purchasing. Generally, the life cycle of IT product is very short, the product specification always changed with new product release.

Although it is easy to get IT product specification form WWW, this kind of information usually disperse in many different web sites. Traditionally, to collect the IT product specification such as desktop computer, one needs to search the related web pages by using search engines, either yahoo or google, then browse these web pages manually to collect the information we need. It takes large effort for information collection.

Most of product specifications are described with specific format and embedded in web page. Due to browser or software agent can't identify the production specification form natural language document, it cause the simple job like "product specification collection" can't be automated. The advance in natural language process technology makes it is possible to extract specific information piece from web page that describe in natural language.

In other hand, the concept of Semantic Web and ontology provide a new model for information presentation. Through the ontology, information could be produce, exchange or analysis by automatic process to enhance the efficiency of knowledge management.

So that, our research try to build up a automatic process that link the natural language process technology and ontology concept, and apply this process to extract IT product specification form Web page. The objectives of research are show as following:

1. Build up a prototype system for automatic IT product specification extraction that can extract information formation form web page efficiently and accurately
2. Save the extracted information in ontology language to provide widely

application and push the developing of Semantic Web.

1.3 Research Methodology and progress

We build up a prototype system to research the automatic information extraction process that combined with natural language process and ontology concept. Our research steps are:

1. Define the research topic, objective and scope:

   At first, we define the research topic, objective and scope to guide the whole research progress.

2. Literature review:

   Base on research topic, objective and scope, one of major task is to build up a prototype to extract IT product specification information. So we must select an extraction methodology to meet our requirement.

   We review the literatures about knowledge management, natural language process, information extraction, semantic web and ontology to understand current status of related research and decide use text mining and grammar rule to extract IT product specification information.

3. Build up prototype:

   The construction of prototype could be separated 3 tasks:

   1. NLP tools survey: Review current NLP tools to select a NLP platform
   2. Ontology Developing: Define the scope and relation of IT product specification that we want to extract.
   3. JAPE grammar rule developing: Develop optimal JAPE grammar rule for IT product specification.

4. Analysis and evaluation:

   We select and download 36 web page for system testing and evaluate the extraction performance.

5. Conclusion:

   Base on the extraction result and performance evaluation, we summarize the advantage and disadvantage of rule base extraction methodology.

Figure 1 show the research flow and progress.

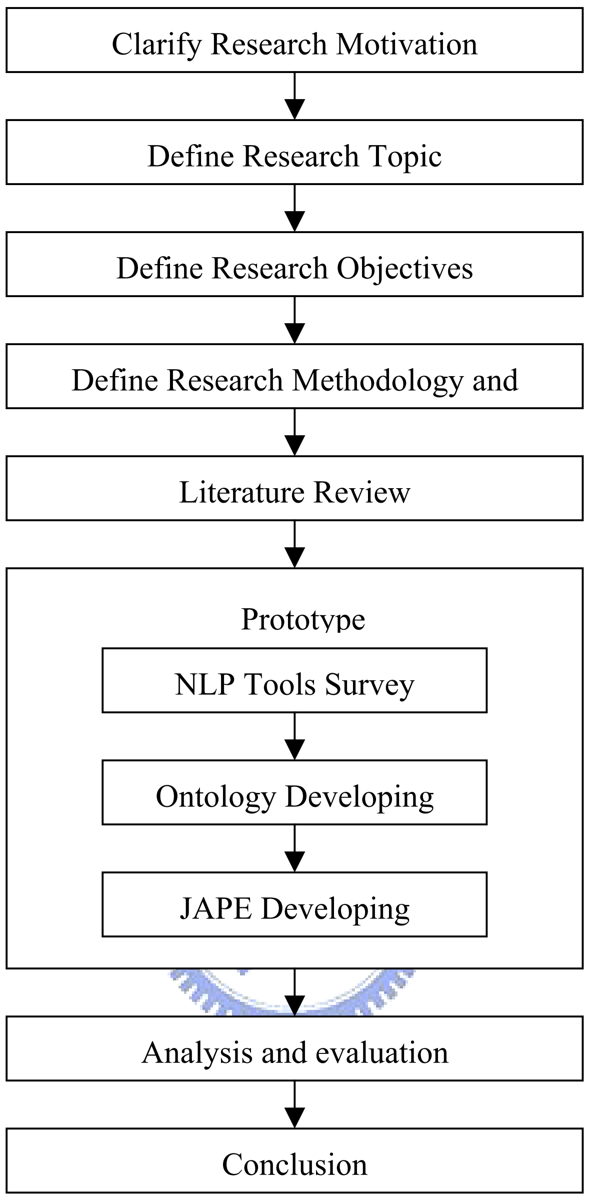


Figure 1: Research flow and progress

1.4 Research Scope and Limitation

To extract specification information from Web page, we have to developing information extraction pattern for specific domain knowledge. The IT product is a wide set that difficult to develop a general pattern for all IT product, so we select part of IT product for prototype developing. The target IT product as following:

- Personal Computer
- Unix Server
- Monitor
- Printer

Although product specification has common format, the little difference are exist in different web site. To enhance the precision of information extraction, we aim HP and IBM web site as target template to develop optimize information extraction rule for IT product of HP and IBM. The information extraction rule also can apply to other Web site, but the precision of information extraction maybe down. We will compare and discuss this issue in chapter 5.

2. Literature review

2.1 Natural Language Processing

Natural language processing (NLP) is the area of study that focuses on techniques that enable machines to work with human language. This involves not only the "understanding" or analysis of language, but also the generation or production of language [7].

A "natural language" (NL) is any of the languages naturally used by humans, i.e. not an artificial or man-made language such as a programming language. The "Natural language processing" (NLP) is a convenient description for all attempts to use computers to process natural language [8]. NLP includes:

- Speech synthesis:

  Although this may not at first sight appear very 'intelligent', the synthesis of natural-sounding speech is technically complex and almost certainly requires some 'understanding' of what is being spoken to ensure, for example, correct intonation.

- Speech recognition:

  Recognize of continuous sound waves to discrete words.

- Natural language understanding:

  Here treated as moving from isolated words (either written or determined via speech recognition) to 'meaning'. This may involve complete model systems or 'front-ends', driving other programs by NL commands.

- Natural language generation:

Generating appropriate NL responses to unpredictable inputs.

- Machine translation (MT):

  Translating one NL into another.

NLP for information extraction has been started for a long time. In 1992, Tomek Strzalkowski and Barbara Vautheyl tried to build up a prototype of information retrieval system which uses advanced natural language processing techniques to enhance the effectiveness of traditional key-word based document retrieval [9]. The information retrieval system consists of a traditional statistical backbone (Harman and Candela, 1989) augmented with various natural language processing components that assist the system in database processing and translate a user's information request into an effective query.

To enhance the information extraction in natural language processing, Cynthia A. Thompson's research team try to apply the active learning to reduce annotation effort in 1999. They developed a system that learns rules for information extraction. The goal of an IE system is to find specific pieces of information in a natural-language document [10].

2.2 Information Extraction

Information extraction is the task of converting documents containing fragments of structured information embedded in other extraneous material into a structured template or database-like representation [3]. Since WWW is a large information repository, our major concern is the approach to extract information for web document.

Claire Cardie had provided a architecture for information extraction system in 1997. The architecture defines 5 steps to extract information from natural language document

[11].

1. Tokenization and Tagging:

   Each input text is first divided into sentences and words in a tokenization and tagging step.

2. Sentence Analysis:

   It comprises one or more stages of syntactic analysis, or parsing, that together identify noun groups, verb groups, prepositional phrases, and other simple constructs.

3. Extraction:

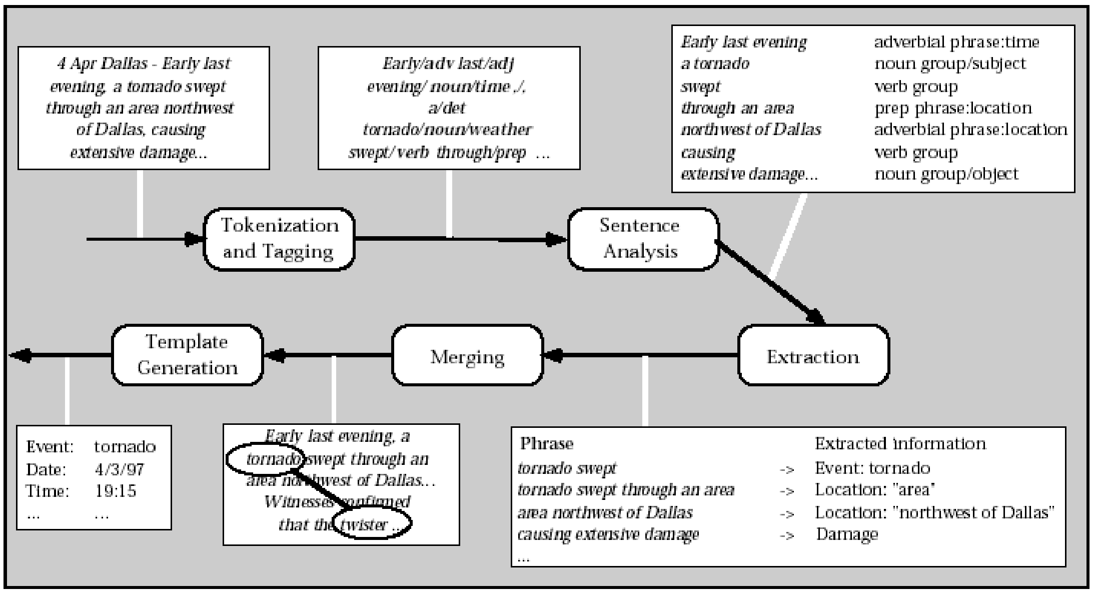
   The extraction phase is the first entirely domain specific component of the system. During extraction, the extraction phase identifies domain specific relations among relevant entities in the text.

4. Merging:

   The main job of the merging phase is co-reference resolution, or anaphora resolution: The system examines each entity encountered in the text and determines whether it refers to an existing entity or whether it is new and must be added to the system's discourse-level representation of the text.

5. Template generation:
   The template generation phase determines the number of distinct events in the text, maps the individually extracted pieces of information onto each event and produces output templates.

   The process flow is show as figure 2:



4 Apr Dallas - Early last evening, a tornado swept through an area northwest of Dallas, causing extensive damage...
Early/adv last/adj evening/ noun/time ./. a/det tornado/noun/weather swept/verb through/prep ...
Early last evening — adverbial phrase:time
a tornado — noun group/subject
swept — verb group
through an area — prep phrase:location
northwest of Dallas — adverbial phrase:location
causing — verb group
extensive damage... — noun group/object
Tokenization and Tagging
Sentence Analysis
Extraction
Merging
Template Generation
Event: tornado
Date: 4/3/97
Time: 19:15
... ...
Early last evening, a tornado swept through an area northwest of Dallas... Witnesses confirmed that the twister ...
Phrase — Extracted information
tornado swept -> Event: tornado
tornado swept through an area -> Location: "area"
area northwest of Dallas -> Location: "northwest of Dallas"
causing extensive damage -> Damage
...

Figure 2: Architecture for an Information-Extraction System

Generally, there are tow major approaches to extract information form WWW, one is text mining and another is Web mining.

2.2.1 Text Mining

Text mining should not be confused with the better known Internet search engine tools or database management capabilities. Analogous to data mining, which extracts useful information from any type of data with large quantities, text mining is a procedure applied to large volumes of free unstructured text. After a traditional search for documents is completed, such as in format of full text, abstracts, or indexed terms, text mining explores the complex relationship among documents [12].

Text mining is about looking for patterns in natural language text, and may be defined as the process of analyzing text to extract information from it for particular purposes. There are three major concerns about text mining [13,14,15]:

(1) Information Retrieval, the foundational step of text mining. It is the extraction

of relevant records from the source technical literatures or text databases for further processing.

(2) Information Processing, the extraction of patterns from the retrieved data obtained in the previous step. According to Kostoff, it has three components: bibliometrics, computational linguistics and clustering techniques. This step typically provides ordering, classification and quantification to the formerly unstructured material.

(3) Information Integration. It is the combination of the information processing computer output with the human cognitive processes.

Raymond J. Mooney and Un Yong Nahm present a framework for text mining based on the integration of Information Extraction (IE) and Knowledge Discovery in 2002 [16].

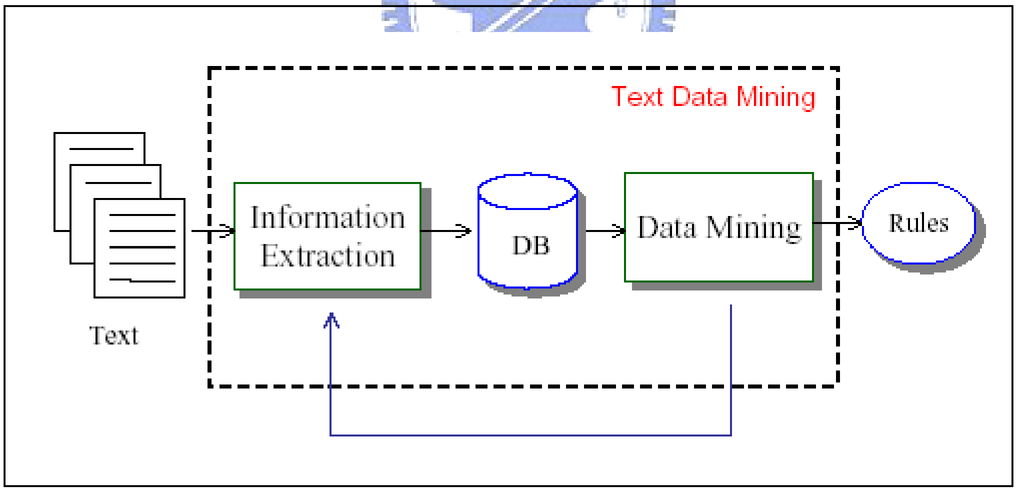


Figure 3: The overview of IE base text mining framework

They use the application of data mining techniques to automated discovery of useful or interesting information from unstructured text. Several techniques have been proposed for text mining, including conceptual structure, association rule mining, episode rule mining, decision trees, and rule induction methods.

Claire Grover's research team provides another methodology for test mining in 2004. They propose a framework for text mining services that apply NLP tools to annotate XML document and extract information from natural language document. The Workflow involves four major steps:

1. Tokenization: Identifying and marking up words and sentences in the input text.

2. Location Tagging with a classifier: Using a trained maximum entropy classifier to mark up location names.

3. Location Tagging by Lexicon: Using a lexicon of location names to mark up additional locations not identifier by the tagger.

4. Gazetteer Query: Sending location names extracted from the text to a gazetteer resource, and presenting the query results in an application-appropriate form.

Weiguo Fan's research team describes a generic process model for a text mining application in 2005. Their process starting with a collection of documents, a text mining tool would retrieve a particular document and preprocess it by checking format and character sets. Then it would go through a text analysis phase, sometimes repeating techniques until information is extracted. Three text analysis techniques are shown in the example, but many other combinations of techniques could be used depending on the goals of the organization. The resulting information can be placed in a management information system, yielding an abundant amount of knowledge for the user of that system [17].

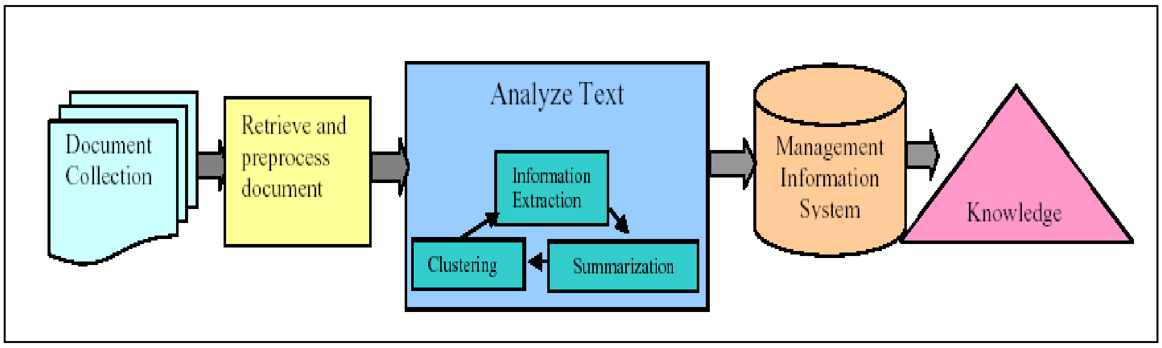


Figure 4: The process flow for text mining

2.2.2 Web mining

Two important and active areas of current research are data mining and the World Wide Web. The web mining is the combination of these two areas, has been the focus of several recent research projects and papers.

Web mining is the use of data mining techniques to automatically discovery and extract information from Web document and services. Web mining can be broadly defined as the discovery and analysis of useful information from the World Wide Web. This broad definition on the one hand describes the automatic search and retrieval of information and resources available from millions of sites and on-line databases, i.e., Web content mining, and on the other hand, the discovery and analysis of user access patterns from one or more Web servers or on-line services, i.e., Web usage mining.

The taxonomy of Web mining along its two primary dimensions, namely Web content mining and Web usage mining. We also describe and categorize some of the recent work and the related tools or techniques in each area. This taxonomy is depicted in Figure 5 [18]

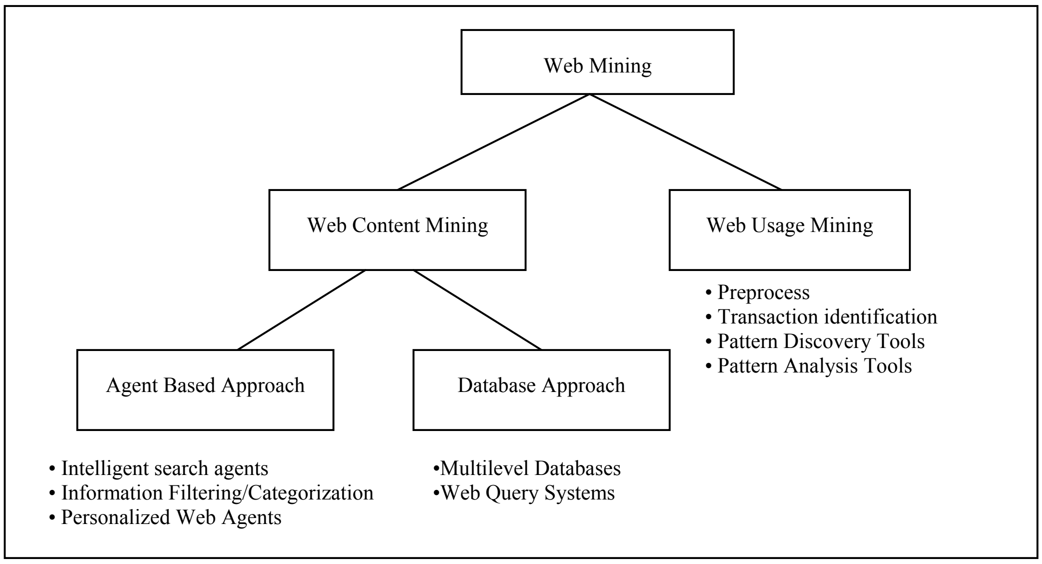


Figure 5: Taxonomy of Web mining

To mining web content, agent-base tools and database mining techniques are two major approaches. Agent base approach to Web mining involves the development of sophisticated AI systems that can act autonomously or semi-autonomously on behalf of a particular user, to discover and organize Web-based information. The database approaches to Web mining have generally focused on techniques for integrating and organizing the heterogeneous and semi-structured data on the Web into more structured and high-level collections of resources, such as in relational databases, and using standard database querying mechanisms and data mining techniques to access and analyze this information.

2.3 Semantic Web and Ontology

2.3.1 Semantic Web

The Semantic Web is the representation of data on the World Wide Web. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming.

Semantic Web tries to build a universal schema to unify the different knowledge schemas on the web. Semantic Web proposes an architecture that consists of multiple layers to construct a universal schema framework. The figure 6 shows the layer structure of Semantic Web.

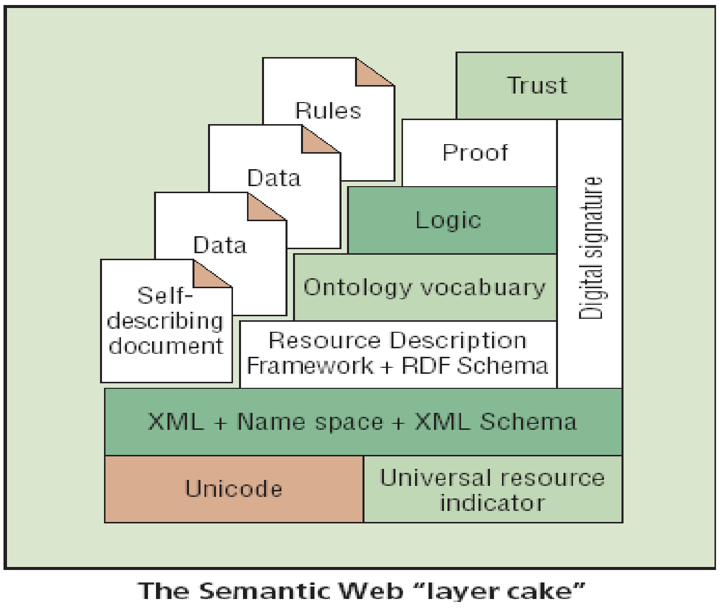

Trust
Rules
Proof
Data
Data
Logic
Digital signature
Self-describing document
Ontology vocabuary
Resource Description Framework + RDF Schema
XML + Name space + XML Schema
Unicode
Universal resource indicator
The Semantic Web "layer cake"

Figure 6: The layer of Semantic Web

The basis of the architecture is RDF. It provides a special format for every semantic statement on the web: (Subject, Predicate, Object). It is the basic syntax for the whole web; that is, all programs can recognize this format.

RDF is the first step (and hence the basis) of the Semantic Web architecture. But, constructing a universal schema for the web is not easy. RDF itself can't be the universal schema because the semantics of data encoded in it are not specified yet. Hence there are RDF Schema, Ontology, and Rules Layers that help to specify the meanings of the subjects, predicates, and objects used in RDF statements. Together they can precisely define the semantics of RDF statements. Finally the Logic Framework Layer is needed to define the working mechanism of machines and these portions. Search on Semantic Web should follow this mechanism. Using Semantic Web's approach, all machines can understand data on the web by first recognizing RDF statements, and finding the semantic definitions in the specified URIs (they may be linked to RDF Schema documents, ontology documents, or documents about rules), then the machines can perform search or provide services more smoothly since they can really recognize all the data on the web. There will be only one schema on the web then.

2.3.2 Ontology

Ontology is an explicit formal specification of how to represent the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. An ontology define the common words and concepts that used to describe and represent an area of knowledge. Ontology models the vocabulary and meaning of domains of interests in a computer-usable form that computer can understand and share domain knowledge for each other.

We can now clarify the role of an ontology that considered as a set of logical axioms designed to account for the intended meaning of a vocabulary. Given a language L with ontological commitment K, an ontology for L is a set of axioms designed in a way such that the set of its models approximates as best as possible the set of intended models of L according to K (see figure 7). In general, it is not easy to find the right set of axioms, so that an ontology will admit other models besides the intended ones. Therefore, an ontology can "specify" a conceptualization only in a very indirect way, since

(i) It can only approximate a set of intended models;

(ii) Such a set of intended models is only a weak characterization of a conceptualization.

We shall say that an ontology O for a language L approximates a conceptualization C if there exists an ontological commitment **K** = <**C**, $\Im$> , where **C** = <D, W, $\Re$> is a conceptualization and $\Im: V \rightarrow D \cup \Re$ is a function assigning elements of D to vocabulary V, and elements of $\Re$ to predicate vocabulary V. The symbol D is a domain and W is a set of relevant states of affairs of such domain and Â is a set of conceptual relations on <D, W>. Such that the intended models of L according to K are included in the models of O .

An ontology commits to C if:

(iii) It has been designed with the purpose of characterizing C, and

(iv) It approximates C.

A language L commits to an ontology O if it commits to some conceptualization C such that O agrees on C. With these clarifications, we come up to the following definition, which refines Gruber's definition by making clear the difference between an ontology and a conceptualization [19]:

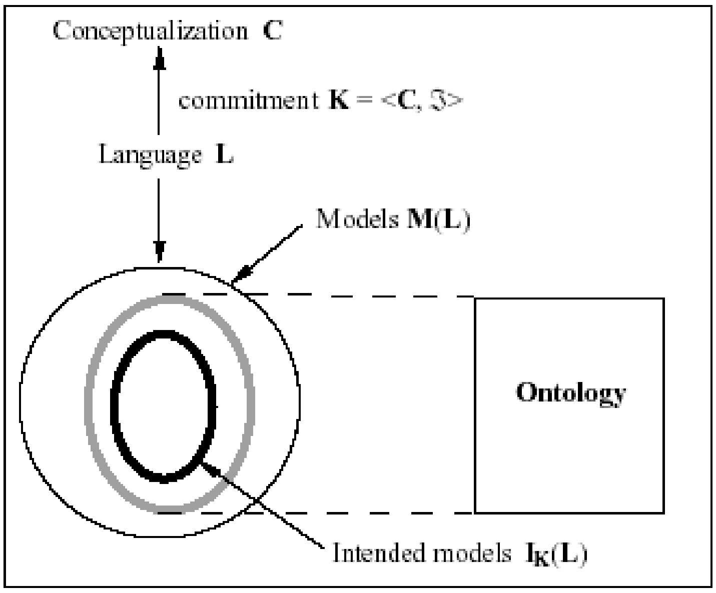


Figure 7: The intended models of a logical language reflect its commitment to a conceptualization

An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models.

The relationships between vocabulary, conceptualization, ontological commitment and ontology are illustrated in Figure 2. It is important to stress that an ontology is language-dependent, while a conceptualization is language-independent.

2.3.3 Ontology Language

Several ontology languages have been developed during the last few years, and they will surely become ontology languages in the context of the Semantic Web. Some of them are based on XML syntax, such as Ontology Exchange Language (XOL), SHOE (which was previously based on HTML), and Ontology Markup Language (OML), whereas Resource Description Framework (RDF) and RDF Schema are languages created by Word Wide Web Consortium (W3C) working groups. Finally, two additional languages are being built on top of RDF(S), the union of RDF and RDF Schema to improve its features: Ontology Inference Layer (OIL) and DAML+OIL. Other languages have also been used, traditionally, for building ontologies, but that analysis is out of the scope of this article.

● **XML-based Ontology Exchange Language (XOL)**

The US bioinformatics community designed XOL for the exchange of ontology definitions among a heterogeneous set of software systems in their domain. Researchers created it after studying the representational needs of experts in bioinformatics. They selected Ontolingua and OML as the basis for creating XOL, merging the high expressiveness of OKBC-Lite, a subset of the Open Knowledge Based Connectivity protocol, and the syntax of OML, based on XML. There are no tools that allow the development of ontologies using XOL. However, since XOL files use XML syntax, we can use an XML editor to author XOL files

● **Simple HTML Ontology Extension (SHOE)**

SHOE, developed at the University on Maryland and used to develop OML, was created as an extension of HTML, incorporating machine-readable semantic knowledge in HTML documents or other Web documents [20]. Recently, the University of Maryland has adapted the SHOE syntax to XML. SHOE makes it possible for agents to gather meaningful information about Web pages and documents, improving search mechanisms, and knowledge gathering. This process consists of three phases: Define an ontology, annotate HTML pages with ontological information to describe themselves and other pages, and have an agent semantically retrieve information by searching all the existing pages and keeping information updated. The Knowledge Annotator annotates ontological information in HTML pages.

● **Ontology Markup Language (OML)**

OML, developed at the University of Washington, is partially based on SHOE. In fact, it was first considered an XML serialization of SHOE. Hence, OML and SHOE share many features.

Four different levels of OML exist: OML Core is related to logical aspects of the language and is included by the rest of the layers; Simple OML maps directly to RDF(S); Abbreviated OML includes conceptual graphs features; and Standard OML is the most expressive version of OML. We selected Simple OML, because the higher layers don't provide more components than the ones identified in our framework. These higher layers are tightly related to the representation of conceptual graphs.

There are no other tools for authoring OML ontologies other than existing general-purpose XML edition tools

● **Resource Description Framework (RDF) and RDF Schema (RDFS)**

RDF, developed by the W3C for describing Web resources, allows the specification of the semantics of data based on XML in a standardized, interoperable manner. It also provides mechanisms to explicitly represent services, processes, and business models, while allowing recognition of non-explicit information.

The RDF data model is equivalent to the semantic networks formalism. It consists of three object types: resources are described by RDF expressions and are always named by URIs plus optional anchor IDs; properties define specific aspects, characteristics, attributes, or relations used to describe a resource; and statements assign a value for a property in a specific resource.

The RDF data model does not provide mechanisms for defining the relationships between properties (attributes) and resources. This is the role of RDFS. RDFS offers primitives for defining knowledge models that are closer to frame-based approaches. RDF(S) is widely used as a representation format in many tools and projects, such as Amaya, Protégé, Mozilla, SilRI, and so on.

● **Ontology Interchange Language (OIL)**

OIL, developed in the OntoKnowledge project (www.ontoknowledge.org/OIL), permits semantic interoperability between Web resources. Its syntax and semantics are based on existing proposals (OKBC, XOL, and RDF(S)), providing modeling primitives commonly used in frame-based approaches to ontological engineering (concepts, taxonomies of concepts, relations, and so on), and formal semantics and reasoning support found in description logic approaches (a subset of first order logic that maintains a high expressive power, together with decidability and an efficient inference mechanism).

OIL, built on top of RDF(S), has the following layers: Core OIL groups the OIL primitives that have a direct mapping to RDF(S) primitives; Standard OIL is the complete OIL model, using more primitives than the ones defined in RDF(S); Instance OIL adds instances of concepts and roles to the previous model; and Heavy OIL is the layer for future extensions of OIL.

OILEd, Protégé2000, and WebODE can be used to author OIL ontologies. OIL's syntax is not only expressed in XML but can also be presented in ASCII.

● **DARPA Agent Markup Language (DAML) + OIL**

DAML+OIL has been developed by a joint committee from the US and the European Union (IST) in the context of DAML, a DARPA project for allowing semantic interoperability in XML [21]. Hence, DAML+OIL shares the same objective as OIL.

DAML+OIL is built on RDF(S). Its name implicitly suggests that there is a tight relationship with OIL. It replaces the initial specification, which was called DAML-ONT, and was also based on the OIL language. OILEd, OntoEdit, Protégé2000, and WebODE are tools that can author DAML+OIL ontologies.

2.4 Ontology-Base knowledge management

About ontology base knowledge extraction process, Harith Alani's research team takes Artequakt project which seeks to automatically extract knowledge about artists from web, populate a knowledge base, and use it to generate personalized narrative biographies [22]. The system architecture of Artequakt system is composed by three key sub-systems that include "knowledge extraction", "Information management" and "Narrative Generation". Figure 8 shows the architecture of Artequakt system.

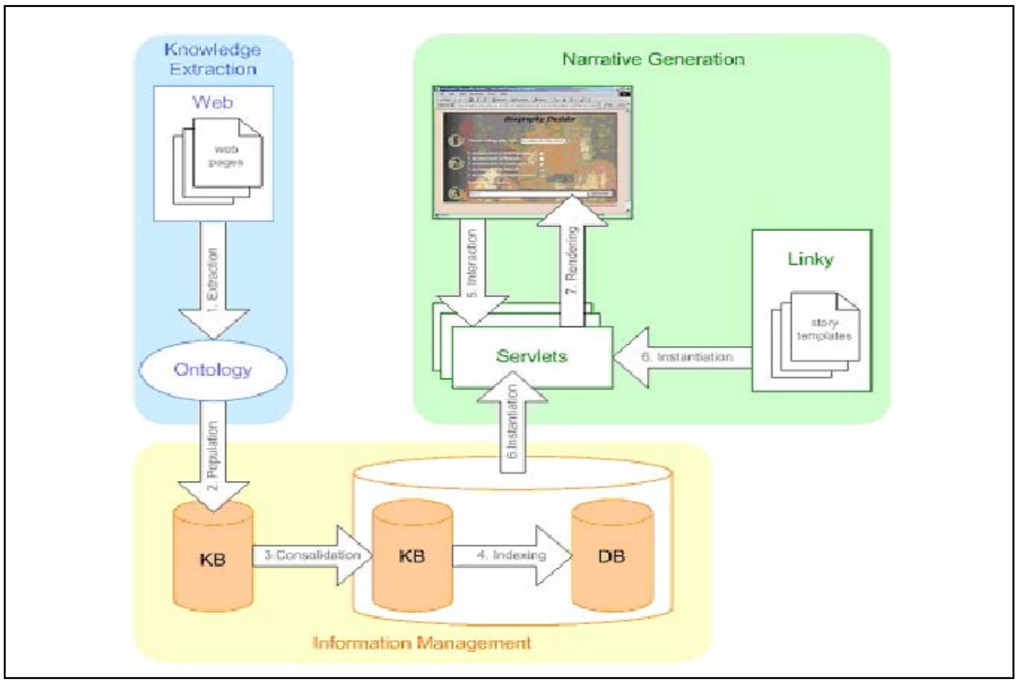


Figure 8: The Artequakt Architecture

The system use ontology and lexical tools to identifying knowledge fragment from web page. The fragments of information are passed to the ontology server along with metadata derived from the vocabulary of the ontology. Artequakt system stores the information by the ontology server and consolidated into a knowledge base that can be queried via an inference engine. The final sub-system is the narrative generation. The Artequakt server takes requests from simple Web interface and generates biography of artist base on user's requirement.

David Vallet, Miriam Fernández, and Pablo Castells also provided an Ontology-Based Information Retrieval Model in 2004. They propose an ontology-based retrieval model for the exploitation of full-fledged domain ontology and knowledge bases, to support semantic search in document repositories. In David's research team view, semantic retrieval problem is very close to the latest proposals in knowledge information management. While KIM focuses on automatic population and annotation of documents, their work focuses on the ranking algorithms for semantic search. [23]

The approach of ontology-based information retrieval can be seen as an evolution of classic keyword-based retrieval techniques. The keyword-based index is replaced by a semantic knowledge base. The overall retrieval process is illustrated in figure 9.

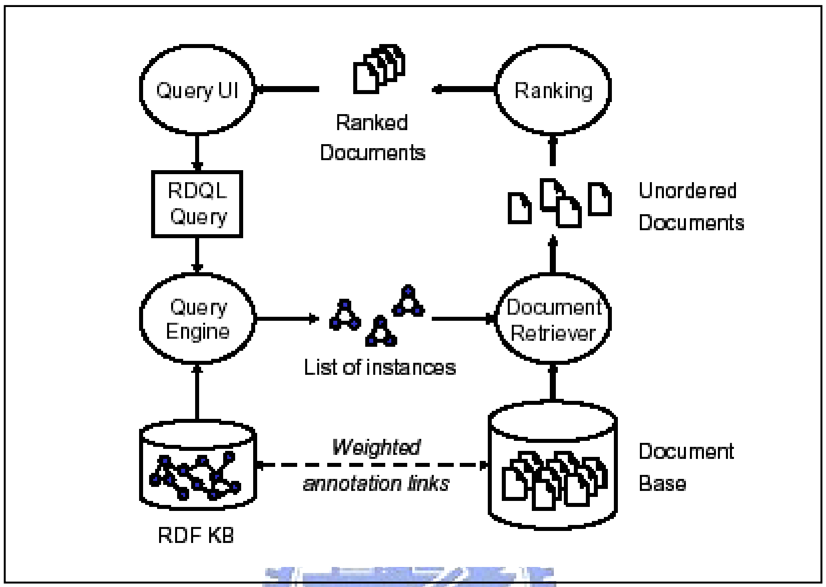


Figure 9: The overview of ontology-base information retrieval

The system takes as input a formal RDQL query. This query could be generated from a keyword query, as in e.g. a natural language query. The RDQL query is executed against the knowledge base, which returns a list of instance that meet the query. Finally, the documents that are annotated with these instances are retrieved, ranked, and presented to the user.

3. System architecture and work flow

3.1 Developing environment

3.1.1 System configuration

We developed IT specification extraction system under Microsoft platform. The hardware and software environment are show as following:

Hardware:
- CPU: AMD Athlon 1G
- Memory: 256 MB DRAM
- Hard disk: 80GB

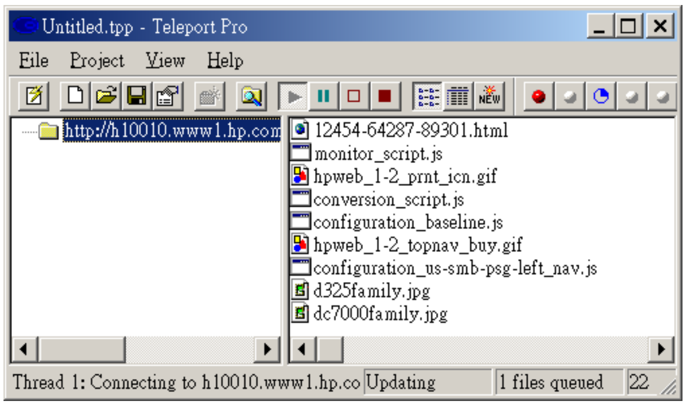
OS: Windows 2000

Software Tools:
- Web page collector: Teleport Pro Ver. 1.29
- NLP developing platform: GATE 3.0 build 1846
- Ontology Editor: SemTalk Ver 1.2.5

NLP tools (GATE 3.0) need much computing resource for natural language processing, especially the memory size. Our hardware configuration just allow a corpus that includes 9~11 web documents, depend on total tokens amount, for processing at same time. It is an unexpected limitation. Due to documents limitation of corpus, we separate IT specification extraction system into 4 applications. Please refer to chapter 3.2 for detail system architecture.

3.1.2 Web page collector: Teleport Pro

Teleport Pro is an offline browsing tool for getting data from the Internet. Input the URL, teleport can download all component of web page and save into local hard disk.



http://h10010.www1.hp.com

12454-64287-89301.html
monitor_script.js
hpweb_1-2_prnt_icn.gif
conversion_script.js
configuration_baseline.js
hpweb_1-2_topnav_buy.gif
configuration_us-smb-psg-left_nav.js
d325family.jpg
dc7000family.jpg

Thread 1: Connecting to h10010.www1.hp.co | Updating | 1 files queued | 22

Figure 10: Web page collector—Teleport Pro

3.1.3 Natural Language Processing Tools: GATE

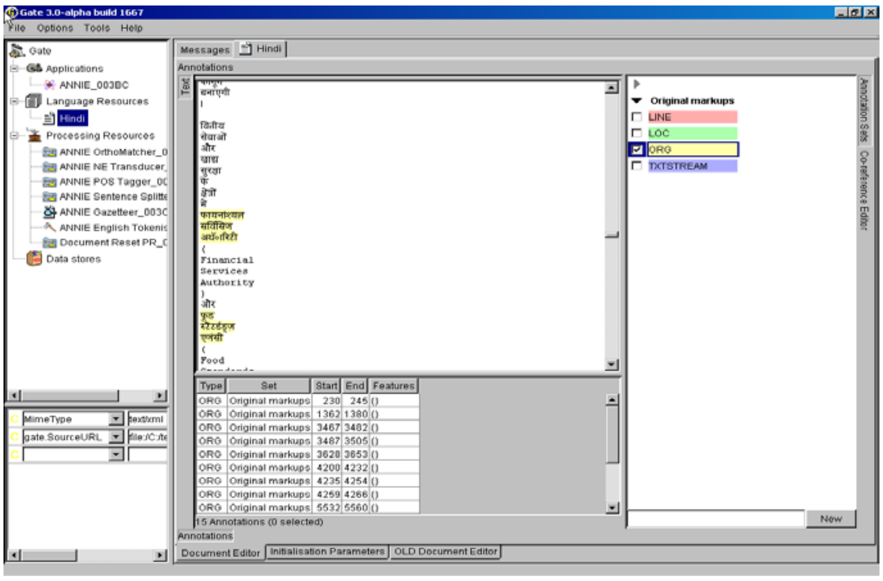
GATE is famous NLP tool that has been built over the past eight years in the Sheffield University NLP group. It comprises an architecture, framework (or SDK) and graphical development environment. The system has been used for many natural language processing projects, in particular for Information Extraction. The system supports the full lifecycle of language processing components, from corpus collection and annotation through system evaluation.

GATE as an architecture suggests that the elements of software systems that process natural language can usefully be broken down into various types of component. Components are reusable software chunks with well-defined interfaces (see figure 11), and are a popular architectural form, used in Sun's Java Beans and Microsoft's .Net, for example. GATE components are specialized types of Java Bean, and come in three Resources [24]:

- Language Resources (LRs) represent entities such as lexicons, corpora or ontologies;
- Processing Resources (PRs) represent entities that are primarily algorithmic, such as parsers, generators or ngram modellers;

- Visual Resources (VRs) represent visualisation and editing components that participate in GUIs.

When using GATE to develop language process functionality for an application, the developer uses the development environment and the framework to construct resources of the three types. This may involve programming, or the development of Language Resources such as grammars that are used by existing Processing Resources, or a mixture of both. The development environment is used for visualization of the data structures produced and consumed during processing, and for debugging, performance measurement and so on.



MimeType	text/xml
gate.SourceURL	file:/C:/te

```
बनाएगी
।

वित्तीय
सेवाओं
और
खाद्य
सुरक्षा
के
क्षेत्रों
में
फायनान्शियल
सर्विसिज
अथॉरिटी
(
Financial
Services
Authority
)
और
फूड
स्टैंडर्ड्स
एजेंसी
(
Food
```

Type	Set	Start	End	Features
ORG	Original markups	230	245	()
ORG	Original markups	1362	1380	()
ORG	Original markups	3467	3482	()
ORG	Original markups	3487	3505	()
ORG	Original markups	3628	3653	()
ORG	Original markups	4200	4232	()
ORG	Original markups	4235	4254	()
ORG	Original markups	4259	4266	()
ORG	Original markups	5532	5560	()

15 Annotations (0 selected)

Figure 11:The user interface of GATE

GATE can be used for many things, but one of the most typical uses is to annotate pages with it. This means that we have a collection of pages and a number of concepts (Annotation Schema) that supposedly occur in these pages. GATE provides an easy to use interface for indicating which pieces of text denote which of your concepts. GATE also can annotate all HTML tags that find in text page.

3.1.4 JAPE grammar language

JAPE (Java Annotation Patterns Engine) provides finite state transduction over annotations based on regular expressions. JAPE is a version of CPSL - Common Pattern Specification Language. It allows you to recognize regular expressions in annotations on documents. Typically, regular expressions are applied to character strings, a simple linear sequence of items, JAPE applying them to a much more complex data structure. The result is that in certain cases the matching process in non-deterministic, for example, the results are dependent on random factors like the addresses at which data is stored in the virtual machine, when there is structure in the graph being matched that requires more than the power of a regular automaton to recognise, JAPE chooses an alternative arbitrarily.

A JAPE grammar consists of a set of phases, each of which consists of a set of pattern/action rules. The phases run sequentially and constitute a cascade of finite state transducers over annotations. A JAPE rule is combined with LHS (left-hand-side) and RHS (right-hand-side). The LHS of the rules consist of an annotation pattern that may contain regular expression operators. The RHS consists of annotation manipulation statements. Annotations matched on the LHS of a rule may be referred to on the RHS by means of labels that are attached to pattern elements.

For LHS, there are 3 main ways in which the pattern can be specified:

- specify a string of text,

  e.g. {Token.string == "of"}

- specify an annotation previously assigned from a gazetteer, tokeniser, or other module,

  e.g. {Lookup}

- specify the attributes (and values) of an annotation),

  e.g. {Token.kind == number}

The RHS of the rule contains information about the annotation. Information about the annotation is transferred from the LHS of the rule using the label just described, and annotated with the entity type. Finally, attributes and their corresponding values are added to the annotation. Alternatively, the RHS of the rule can contain Java code to create or manipulate annotations

For example, a single rule is sufficient to identify an IP address, because there is only one basic format - a series of numbers, each set connected by a dot. The rule for this is given below:

```
Rule: IPAddress
(
 {Token.kind == number}
 {Token.string == "."}
 {Token.kind == number}
 {Token.string == "."}
 {Token.kind == number}
 {Token.string == "."}
 {Token.kind == number}
)
:ipAddress -->
 :ipAddress.Address = {kind = "ipAddress"}
```

Figure 12: JAPE Grammar rule example

GATE supports ontology aware grammar transduction, this allows a JAPE transducer to match not only those features on the left hand side of a rule that match it exactly, but also to match any features that are subclasses of those specified in the

JAPE rule. For example, if the ontology specifies that a BMW is a car, and that a car is a vehicle, then a rule that specifies vehicle will match when it finds an instance of BMW or car.

3.2 System Architecture

The IT product specification extraction system is composed of web page collection tool and NLP tools. The system architecture and work flow show as figure 13:

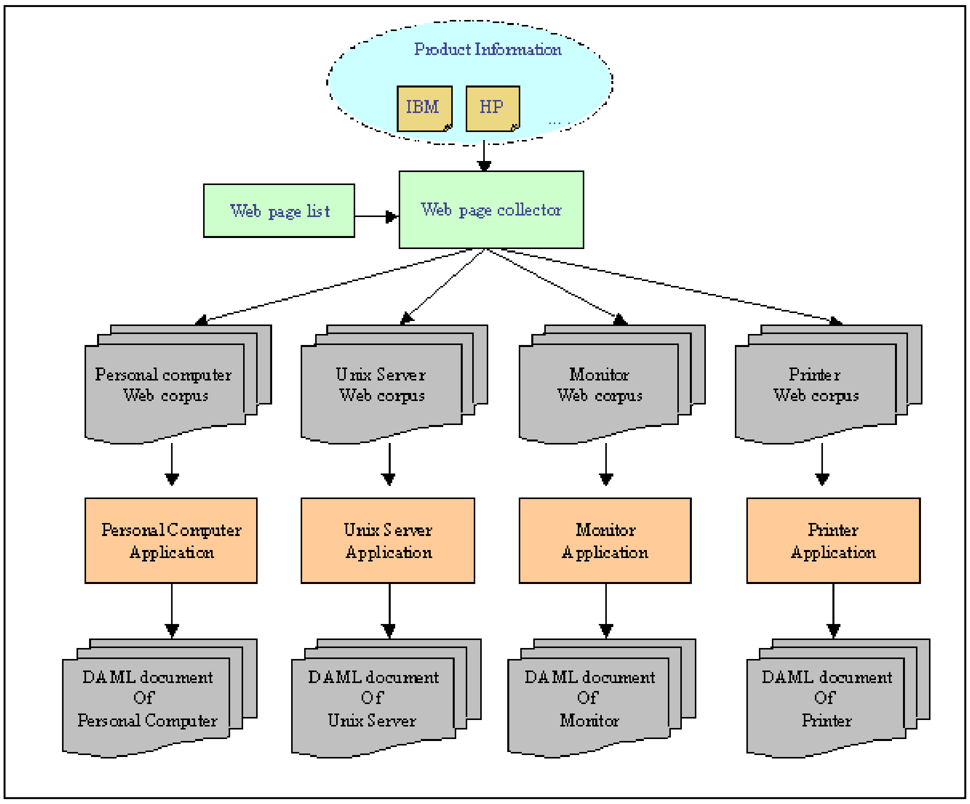


Figure 13: System architecture and workflow

This system can be segmented to 4 IT applications that are "Personal Computer Application", "Unix Server application", "Monitor application" and "Printer application". Each IT application is independent sub-system to extract information from specific knowledge domain. The application has related language resource (such as Web corpus, ontology) and process resource (such as “English Tokeniser”, “Sentence Splitter”). Every application includes 3 major processes that are “Web document annotation”, “JAPE transduction” and “DAML+OIL exporter”. Figure 14 shows the architecture of IT application.

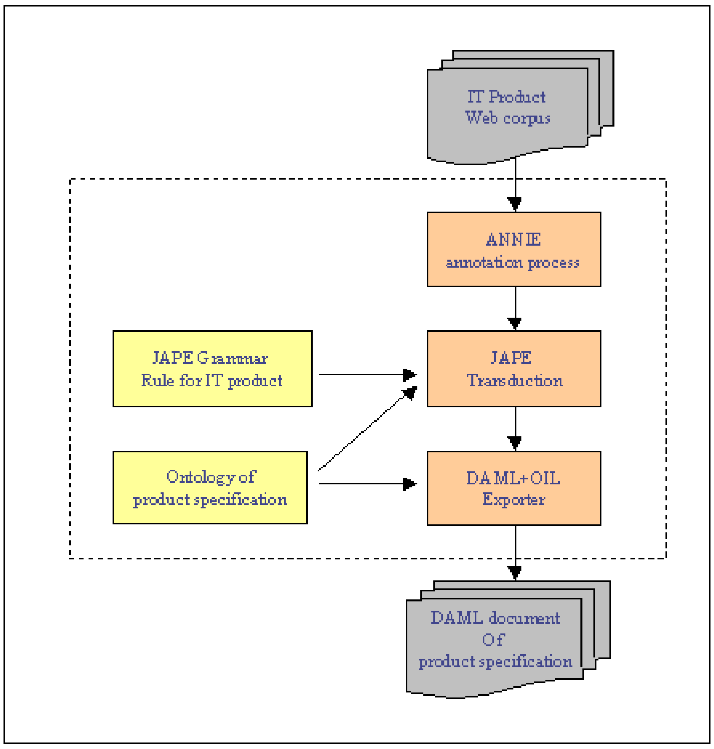


Figure 14: Architecture of IT product application

The system collects web document base on pre-defined web list and save it into specific path as a corpus by different IT product. NLP tools (GATE) will load the corpuses to annotate these documents, than JAPE transducer will load JAPE grammar rule to mark up product specification. Finally, DAM+OIL exporter generate output file of specification in DAML document format.

3.3 Web page collection

The major task of web page collector is download web document base on pre-define web list and save these document in specific path by different IT product. We choose “Teleport Pro” to be the web page collector, Teleport Pro is a widely used offline browsing Web-spider.

To limit the scope, we just down load web page from IBM and HP web site, because IBM and HP have rich product lines to meet our requirement. We also limit the IT product scope to be Unix Server, Desktop, Notebook, Monitor and Printer. We randomly select 2~3 products from HP and IBM per product line, total 34 web pages be down load for prototype system developing. We load the related web documents into GATE as a “corpus”. Corpus is a language resource of GATE that includes several documents for batch process.

3.4 ANNIE annotation process flow

GATE provides a baseline set of reusable and extendable language processing components for common NLP tasks, known collectively as ANNIE (A Nearly New Information Extraction System). ANNIE currently produces precision and recall figures for named entity recognition of around 90%, depending on the text type. [25] ANNIE relies on finite state algorithms and the JAPE language. ANNIE components form a pipeline that show as figure 15:

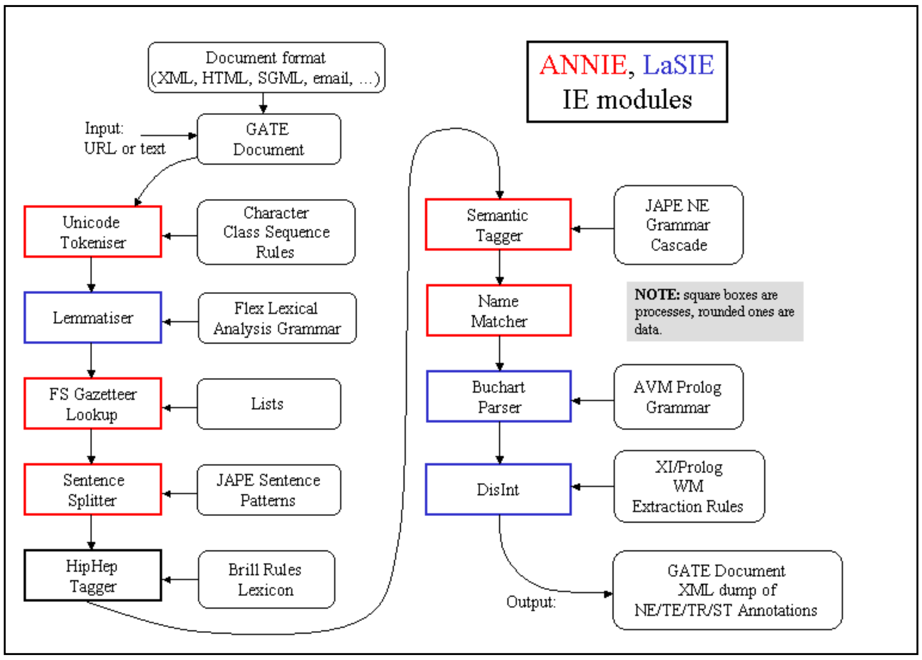
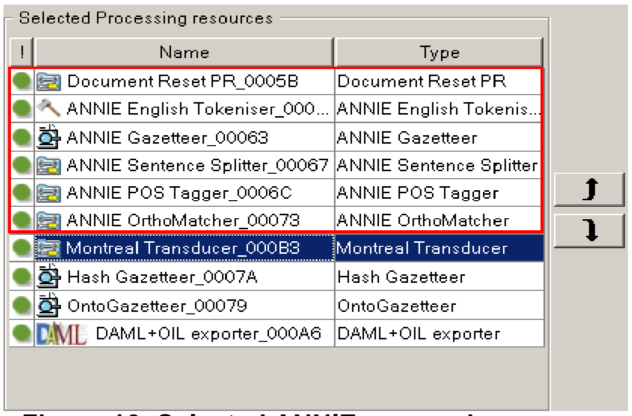


Figure 15: ANNIE process pipeline in GATE 3.0

ANNIE includes many companies like Unicode Tokeniser, Sentence Splitter, POS Tagger, ANNIE Gazetteer, Semantic Tagger, Nominal coreferencer and pronominal coreferencer … and so on. To enhance system performance, we just apply part of component in our prototype system for natural language processing. The ANNIE process flow of IT specification extraction system show as figure 16 and figure 17


Selected Processing resources

!	Name	Type
●	Document Reset PR_0005B	Document Reset PR
●	ANNIE English Tokeniser_000...	ANNIE English Tokenis...
●	ANNIE Gazetteer_00063	ANNIE Gazetteer
●	ANNIE Sentence Splitter_00067	ANNIE Sentence Splitter
●	ANNIE POS Tagger_0006C	ANNIE POS Tagger
●	ANNIE OrthoMatcher_00073	ANNIE OrthoMatcher
●	Montreal Transducer_000B3	Montreal Transducer
●	Hash Gazetteer_0007A	Hash Gazetteer
●	OntoGazetteer_00079	OntoGazetteer
●	DAML+OIL exporter_000A6	DAML+OIL exporter

Figure 16: Selected ANNIE processing resource

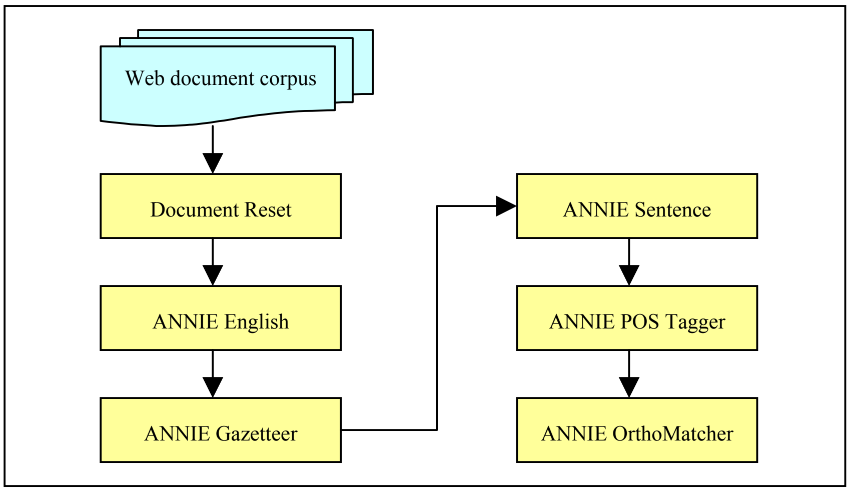


Figure 17: ANNIE process flow in IT specification extraction system

1. **Document Reset:** The document reset resource enables the document to be reset to its original state, by removing all the annotation sets and their contents, apart from the one containing the document format analysis (Original Markups). This resource is normally added to the beginning of an application, so that a document is reset before an application is rerun on that document.

2. **ANNIE English Tokeniser:** The tokeniser splits the text into very simple tokens such as numbers, punctuation and words of different types. The aim is to limit the work of the tokeniser to maximize efficiency, and enable greater flexibility by processing with build-in grammar rules. The English Tokeniser should always be used on English texts that need to be processed afterwards by the POS Tagger.

3. **ANNIE Gazetteer:** The gazetteer lists used are plain text files with one entry per line. Each list represents a set of names, such as names of cities, organizations, days of the week, etc.

4. **ANNIE Sentence Splitter:** The sentence splitter is a cascade of finite-state transducers that segments the text into sentences. This module is required for the tagger. The splitter uses a gazetteer list of abbreviations to help distinguish sentence-marking full stops from other kinds.

5. **ANNIE POS Tagger:** POS Tagger is an external program, passing gate documents as input, and adding some features to the existing Tokens.

6. **ANNIE OrthoMatcher:** The Orthomatcher module adds identity relations between named entities found by the semantic tagger, in order to perform coreference. It does not find new named entities as such, but it may assign a type to an unclassified proper name, using the type of a matching name.

In IT specification extraction system, the major task of ANNIE process is to separate a web document into tokens and add related attributes to these tokens. It is important for following process. We don't expect ANNIE to extract IT specification because the default IE function of ANNIE just can identify "name", "address", "date", "organization". To extract information from specific domain knowledge, customization is necessary. Basically, There are two ways to develop customized extraction rule, one is adding new Gazetteer another is to employ JAPE grammar rules. Consider the flexibility and integration with ontology model, we choose JAPE to be the developing tools.

3.5 Ontology and JAPE extraction rule

The general disadvantage of use NLP technology to extract information is need customized extraction rule for specific knowledge domain. Since build-in IE system can’t extract information from specific know domain, we have to construct extraction rule for IT specification extraction.

As the introduction of section 3.1.4, JAPE (Java Annotation Patterns Engine) provides powerful and flexibility annotation grammar, it also provides ontology aware grammar transduction. In IT product specification extraction system, JAPE transducer is the major process to extract information form specific IT product.

We plan to extract the product specification show as following:

- Personal Computer:
  - Model Name
  - CPU model
  - CPU Speed
  - Memory Type
  - Memory Size
  - Hard Disk Type
  - Hard Disk Size
- Unix Server
  - Server Model
  - CPU Model
  - CPU Speed
  - Cache Memory Size
  - Memory Type
  - Memory Size
  - Max. Memory Size
  - O.S version
  - Hard Disk Type
- Monitor
  - Model Name
  - Monitor Type
  - Monitor Size
  - Pixel Pitch
  - Resolution Mode

Max Resolution Mode

Recommend Resolution Mode

Refresh Rate

Web Price

- Printer

    Model Name

    Print Speed

    Print Quality

    Memory Size

To enable the ontology aware grammar feature, we have to develop ontology for IT product specification. The ontology of IT product specification describes the relation and structure of information that we plan to extract. Figure 18 is an example about ontology of personal computer. In figure 18, Desktop and notebook are sub-class of personal computer, and "Brand", "CPU", "Memory" and "Hard disk" are part of desktop or notebook. "CPU model", "CPU speed"…. and so on are related information of components.

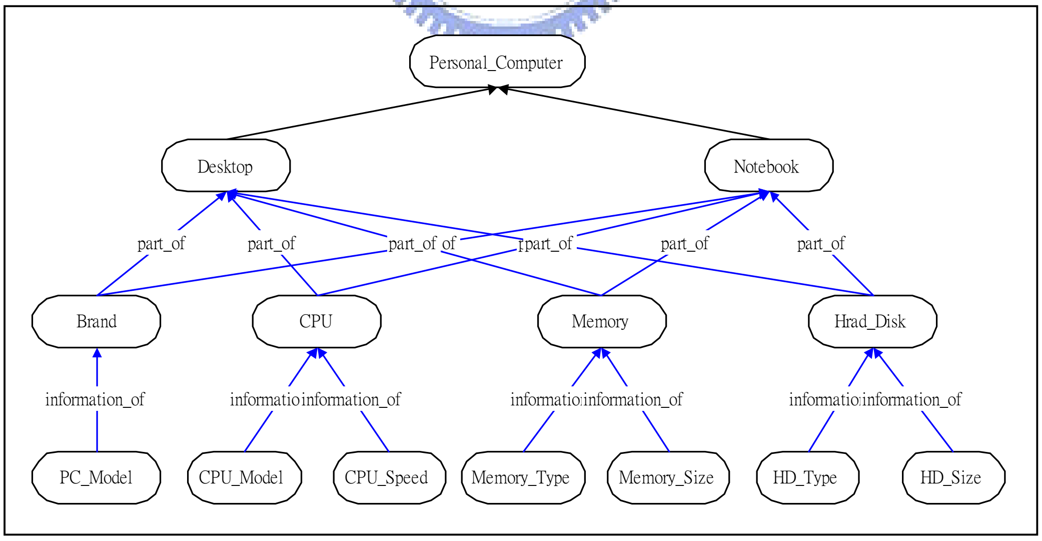


Figure 18: Ontology Example- Ontology of Personal Computer specification

Gate provide a build-in ontology editor for ontology developing, it is a simple developing environment. The build-in ontology just provides “sub-class” relation between parent class and son class. We also tried to create ontology by SemTalk which provide more property and export these ontology to be DAML files. But GATE JAPE transducer seems can’t recognize these properties. Consider the compatibility and precision, we developing ontology with GATE build-in tools. Figure 19 is an example of personal computer ontology developing.

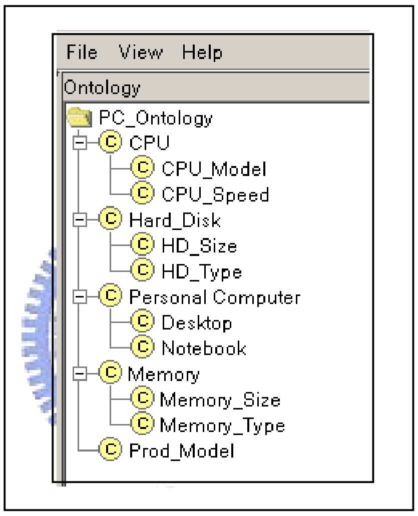


Figure 19: Example of GATE build-in ontology developing

GATE will save the ontology as DAML files. The ontology file must load into GATE as a language resource and refer by JAPE transducer and DAML exporter. Figure 20 is an example of DAML output file, it is ontology of personal computer.

```
<?xml version='1.0'?>
<rdf:RDF
   xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
   xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
   xmlns:daml='http://www.daml.org/2001/03/daml+oil#'>
   <daml:Ontology rdf:about=''
       rdfs:label=''
       rdfs:comment=''>
      <daml:versionInfo>$Id: iswc.daml,v 1.0 2002/04/15 16:51:40 meh Exp $</daml:versionInfo>
   </daml:Ontology>
   <rdfs:Class rdf:about='file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#Memory'
       rdfs:label='Memory'
       rdfs:comment=''/>
   <rdfs:Class rdf:about='file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#Memory_Size'
```

```
        rdfs:label='Memory_Size'
        rdfs:comment=''>

<rdfs:subClassOf>file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#Memory</rdfs:subClassOf>
   </rdfs:Class>
   <rdfs:Class rdf:about='file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#CPU_Model'
        rdfs:label='CPU_Model'
        rdfs:comment=''>

<rdfs:subClassOf>file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#CPU</rdfs:subClassOf>
   </rdfs:Class>
   <rdfs:Class rdf:about='file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#CPU_Speed'
        rdfs:label='CPU_Speed'
        rdfs:comment=''>

<rdfs:subClassOf>file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#CPU</rdfs:subClassOf>
   </rdfs:Class>
   <rdfs:Class rdf:about='file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#HD_Size'
        rdfs:label='HD_Size'
        rdfs:comment=''>

<rdfs:subClassOf>file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#Hard_Disk</rdfs:subClassOf>
   </rdfs:Class>
   <rdfs:Class rdf:about='file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#CPU'
        rdfs:label='CPU'
        rdfs:comment=''/>
   <rdfs:Class rdf:about='file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#Memory_Type'
        rdfs:label='Memory_Type'
        rdfs:comment=''>

<rdfs:subClassOf>file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#Memory</rdfs:subClassOf>
   </rdfs:Class>
   <rdfs:Class rdf:about='file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#Hard_Disk'
        rdfs:label='Hard_Disk'
        rdfs:comment=''/>
   <rdfs:Class rdf:about='file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#HD_Type'
        rdfs:label='HD_Type'
        rdfs:comment=''>

<rdfs:subClassOf>file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#Hard_Disk</rdfs:subClassOf>
   </rdfs:Class>
   <rdfs:Class rdf:about='file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#Prod_Model'
        rdfs:label='Prod_Model'
        rdfs:comment=''/>
  </rdf:RDF>
```

Figure 20: DAML example: Ontology of Personal computer

For performance issue, we develop JAPE grammar rules by difference IT product base on section 1.4. To extract specification information correctly, the customization for JAPE rules to match the text format is necessary. Some times, different web site present same information type in different text format. For example, the CPU cache, IBM web documents show the CPU cache information as “Level1 cache: 4MB” or “L1 cache: 4MB”, but HP web document show as “4MB Level1 cache” or “4MB L1 cache”, it need different JAPE rule to match these information. Figure 21 is an segment of

JAPE code that extract from UNIX server JAPE grammar rule.

```
Rule: CPU_Cache1
(
  (
   {Token.string =~ "[Ll]evel?" } |
   {Token.string =~ "[Ll]?" }
  )
  {Token.kind == number}
  (
    {Token.string == "("}
    {Token.kind == word}
    {Token.kind == number}
    {Token.string == ")"}
  )?
  {Token.string =~ "[Cc]ache?" }
  (
   {Token.kind == punctuation}|
   {Token.kind == word} |
   {Token.kind == number}
  ) ?

  {Token.kind == number}
  (
   {Token.string == "."}
   {Token.kind == number}
  ) ?
  (
   {Token.string == "KB"}   |
   {Token.string == "MB"}
  )
)
:CPU
-->
:CPU.CPU_Cache = {kind = CPU, rule = CPU_Cache1}

Rule: CPU_Cache2
(
  {Token.kind == number}
  (
   {Token.string == "."}
   {Token.kind == number}
  ) ?
  (
   {Token.string == "KB"}   |
   {Token.string == "MB"}
  )
  (
   {Token.string == "shared"} |
   {Token.string == "combined"}
  ) ?
  (
   {Token.string =~ "[Ll]evel?" } |
   {Token.string =~ "[Ll]?" }
  )
  {Token.kind == number}
  {Token.string =~ "[Cc]ache?" }
)
:CPU
-->
:CPU.CPU_Cache = {kind = CPU, rule = CPU_Cache2}
```

Figure 21: Example of JAPE grammar rule segment

The JAPE example uses 2 JAPE rules to extract the information of CPU cache memory. The rule "CPU_cache1" is designed for IBM web document that can match the text format like "Level1 cache: 4MB" or "L1 cache: 4MB". And the rule "CPU_cache2" is designed for HP web document that can match the text format like "4MB Level1 cache" or "4MB L1 cache".

To develop JAPE code for specific web page will enhance the extraction precision, but lost the common usage. We don't expect the JAPE rule CPU_cache1 and CPU_cache2 can extract cache information for the web page of Dell computer. With the extension of web source, the fine tune for JAPE rule to meet new text format is necessary.

3.6 DAML+OIL Exporter

The DAML+OIL Export is a GATE process resource that allows the information segment found in documents to be exported as instances of a specified ontology in DAML+OIL format. When a corpus is processed with ANNIE and JAPE transducer, GATE will mark up the information that we want to extract. When the DAML+OIL exporter processes the corpus, for each information segment found that is of some type (such as CPU_type), if a corresponding concept with the same name as the information type (such as CPU_type) exists in the ontology then a new DAML instance will be generated in the export file. Figure 22 shows the example of DAML output file:

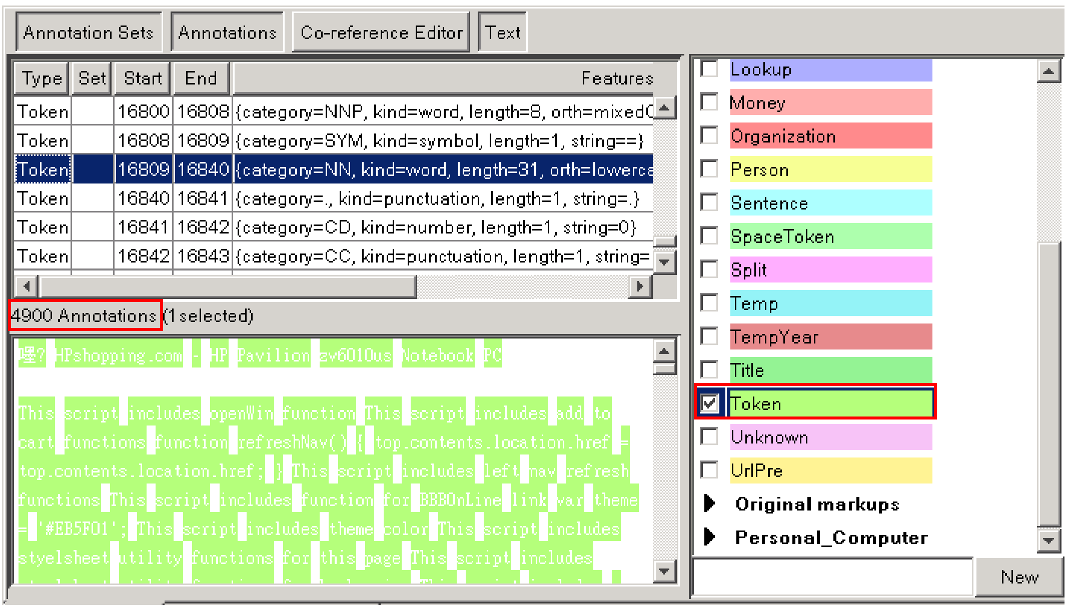
```
<?xml version='1.0'?>
<rdf:RDF
   xmlns:gate='file:/C:/Rick/Thesis/Ontology/Personal_Computer.daml#'
   xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
   xmlns:daml='http://www.daml.org/2001/03/daml+oil#'>
   <daml:Ontology rdf:about=''
       daml:versionInfo='1.0'>
      <daml:comment>autogenerated from GATE RDFFormatExporter </daml:comment>
      </daml:Ontology>

      <daml:Property rdf:about='http://www.daml.org/2001/03/daml+oil#versionInfo'/>

      <daml:Property rdf:about='http://www.daml.org/2001/03/daml+oil#comment'/>

      <gate:Memory_Size rdf:about='512MB'/>

      <gate:HD_Type rdf:about='Enhanced IDE'/>

      <gate:Memory_Size rdf:about='1024MB'/>

      <gate:CPU_Model rdf:about='Pentium簧 M'/>

      <gate:Memory_Size rdf:about='2048MB'/>

      <gate:HD_Size rdf:about='100GB'/>

      <gate:Memory_Type rdf:about='DDR SDRAM'/>

      <gate:Prod_Model rdf:about='Pavilion dv4040us'/>

      <gate:CPU_Speed rdf:about='1.73GHz'/>

</rdf:RDF>
```

Figure 22: Example of DAML output file

4. Analysis and Comparison

4.1 Extraction Result

After annotate processing, web corpus are separated into thousands tokens. Every token has attribute like token type, length, position … and so on. Figure 23 shows the annotation result



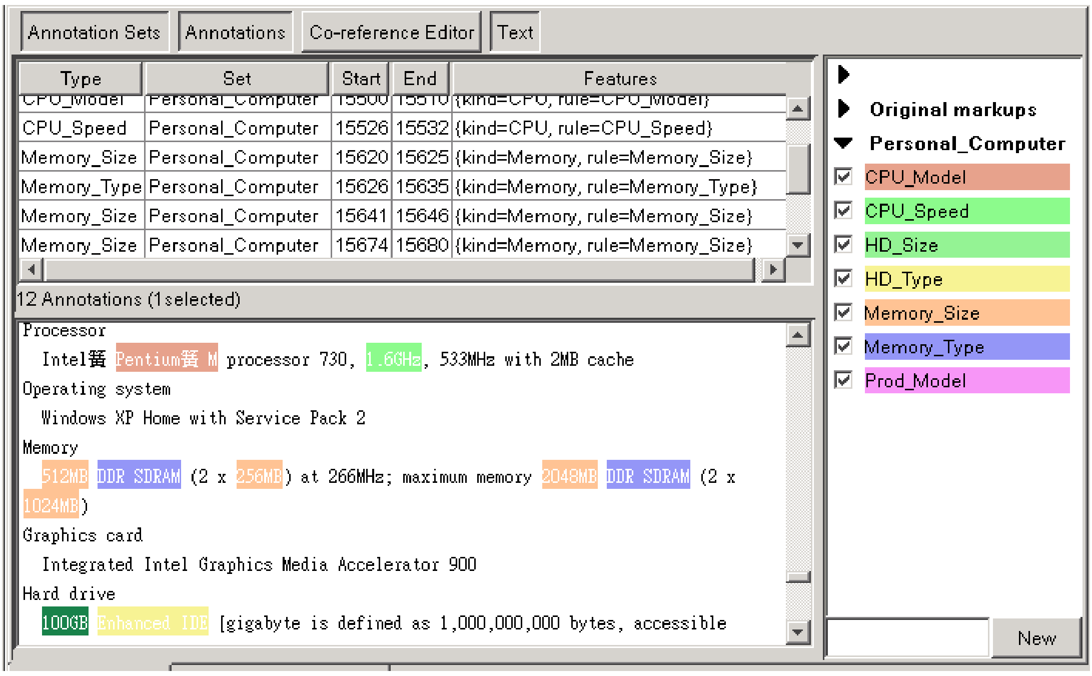
Type	Set	Start	End	Features
Token		16800	16808	{category=NNP, kind=word, length=8, orth=mixedC
Token		16808	16809	{category=SYM, kind=symbol, length=1, string==}
Token		16809	16840	{category=NN, kind=word, length=31, orth=lowerca
Token		16840	16841	{category=., kind=punctuation, length=1, string=.}
Token		16841	16842	{category=CD, kind=number, length=1, string=0}
Token		16842	16843	{category=CC, kind=punctuation, length=1, string=

4900 Annotations (1 selected)

```
哩? HPshopping.com - HP Pavilion zv6010us Notebook PC

This script includes openWin function This script includes add to
cart functions function refreshNav() { top.contents.location.href =
top.contents.location.href; } This script includes left nav refresh
functions This script includes function for BBBOnLine link var theme
= '#EB5F01'; This script includes theme color This script includes
styelsheet utility functions for this page This script includes
```

Figure 23: Annotation result

Then, JAPE transducer tries to match information entity, which may be composed by several tokens, base on JAPE grammar rule. Figure 24 show the process screen that JAPE transducer mark information entity up when the text format is matched with the JAPE grammar rule.



Type	Set	Start	End	Features
CPU_Model	Personal_Computer	15500	15510	{kind=CPU, rule=CPU_Model}
CPU_Speed	Personal_Computer	15526	15532	{kind=CPU, rule=CPU_Speed}
Memory_Size	Personal_Computer	15620	15625	{kind=Memory, rule=Memory_Size}
Memory_Type	Personal_Computer	15626	15635	{kind=Memory, rule=Memory_Type}
Memory_Size	Personal_Computer	15641	15646	{kind=Memory, rule=Memory_Size}
Memory_Size	Personal_Computer	15674	15680	{kind=Memory, rule=Memory_Size}

12 Annotations (1selected)

```
Processor
  Intel籌 Pentium籌 M processor 730, 1.6GHz, 533MHz with 2MB cache
Operating system
  Windows XP Home with Service Pack 2
Memory
  512MB DDR SDRAM (2 x 256MB) at 266MHz; maximum memory 2048MB DDR SDRAM (2 x
1024MB)
Graphics card
  Integrated Intel Graphics Media Accelerator 900
Hard drive
  100GB Enhanced IDE [gigabyte is defined as 1,000,000,000 bytes, accessible
```

Original markups
Personal_Computer
- ✓ 🟥 CPU_Model
- ✓ 🟩 CPU_Speed
- ✓ 🟩 HD_Size
- ✓ 🟨 HD_Type
- ✓ 🟧 Memory_Size
- ✓ 🟦 Memory_Type
- ✓ 🟪 Prod_Model

Figure 24: Information Entity marked up by JAPE transducer

An information entity is one of product specification that may be composed by several tokens, like the monitor resolution mode used to present as “1024x768 @75Mhz”. The goal of IT specification extraction system is to extract these information entities from selected web page. We collect total 32 web pages from IBM and HP web site for system test, the extraction result show as following. The extraction results show that JAPE grammar rule can extract finite information entities form these tokens.

- **Personal computer corpus (include desktop and notebook):**

Table 1: Extraction result of HP and IBM Personal computer

Web Site	Product Name	Total tokens	Extracted information entities
HP	Pavilion dv4030us Notebook	5190	11
HP	Pavilion zv6010us Notebook	4900	10
HP	Media Center zd8110us Notebook	5215	12
HP	Pavilion a1040n Desktop	5104	9
HP	Media Center m7070n Photosmart PC	5103	10
IBM	ThinkPad T42 Notebook	2128	11
IBM	ThinkPad X41Tablet	1806	10
IBM	ThinkPad R50e Notebook	1812	10
IBM	ThinkCentre S50 Desktop	1816	10
IBM	ThinkCentre M51 Desktop	1851	10

- **Unix server corpus**

Table 2: Extraction result of HP and IBM Unix Server

Web Site	Product Name	Total tokens	Extracted Information entities
HP	RX8620	892	18
HP	RX7620	860	16
HP	RX4640	1079	25
HP	RP8420	1120	16
HP	RP7420	1230	18
IBM	p5 590	1016	13
IBM	p5 570	1080	18
IBM	p5 550	1017	11
IBM	p5 520	1031	11
IBM	p5 510	772	11

- **Monitor corpus**

Table 3: Extraction result of HP and IBM monitor

Web Site	Product Name	Total tokens	Extracted Information entities
HP	vs1717 Flat-Panel LCD Monitor	4932	11
HP	f1905 19 LCD Flat-Panel Monitor	4928	11
HP	Pavilion mx704 17 Flat-Screen CRT Monitor	4903	11
HP	s7540 CRT Monitor	527	11
IBM	ThinkVision L170p Monitor	1117	27
IBM	ThinkVision L150 Monito	1050	19
IBM	E74M 17 inch CRT Monitor	1108	25
IBM	E54 15" Monitor MPRII	809	10

-

- **Printer corpus**

Table 4: Extraction result of HP and IBM printer

Web Site	Product Name	Total tokens	Extracted information entities
IBM	Infoprint Color 1357 Printer	782	5
IBM	Infoprint 1412 Printer	807	9
IBM	Infoprint 1352 Printer	889	8
HP	Inkjet 1200dn printer	3766	19
HP	Deskjet 3845 Printer	4905	4
HP	Color LaserJet 2600n Printer	3431	10

We also collect 24 web pages from other company such Sony, ASU, SUN… to compare the extraction result of JAPE rules. The extraction result show as following tables

Table 5: Extraction result of Sony and ASUS personal computer

Web Site	Product Name	Total tokens	Extracted information entities
Sony	Sony VGN-T370P-L Notebooks	3775	10
Sony	Sony VGN-FS675P-H Notebook	3649	8
Sony	Sony VAIO V167G TV-PC Desktop	3534	12
ASUS	ASUS W5A Notebook	549	6
ASUS	ASUS W3V Notebook	663	6
ASUS	ASUS V6V Notebook	531	6

- **Unix server corpus**

Table 6: Extraction result of Sun Unix Server

Web Site	Product Name	Total tokens	Extracted Information entities
SUN	Sun Fire V890 Server	1234	2
SUN	Sun Fire V40z Server	1087	10
SUN	Sun Fire V240 Server	1172	5
SUN	Sun Fire V1280	950	4
SUN	Sun Fire E4900 Server	1285	0
SUN	Sun Fire E20K Server	1209	0

- **Monitor corpus**

Table 7: Extraction result of BenQ and Acer monitor

Web Site	Product Name	Total tokens	Extracted Information entities
BenQ	BenQ USA - FP531 LCD monitor	480	5
BenQ	BenQ FP71V LCD monitor	521	5
BenQ	BenQ FP71E LCD monitor	419	5
BenQ	BenQ FP537s LCD Monitor	467	5
Acer	Acer AF715 CRT monitor	217	4
Acer	Acer AC501 CRT monitor	190	4

-

- **Printer corpus**

Table 8: Extraction result of Epson and Canon Printer

Web Site	Product Name	Total tokens	Extracted information entities
Epson	Epson PictureMate	1458	1
Epson	Epson Stylus Photo R300 Printer	1721	2
Epson	Epson Stylus C66 Printer	1369	3
Canon	CanonPIXMA iP3000 printer	1033	1
Canon	Canon Printers - i80 Printer	1022	2
Canon	Canon PIXMA iP90 Printer	1128	1

4.2 Performance index

Although IT product specification extraction system has the ability to extract information from web pages, we also need some indexes to evaluate the extraction performance. Evaluation of information extraction methods is a very important. There is a general consensus that evaluation is either quantitative or qualitative. Quantitative evaluation measures the performance of the various software algorithms that constitute the extraction tool. Qualitative evaluation assesses the adequacy of information extraction method for specific knowledge domain.[26]

Two evaluation stages are typically performed when evaluating an information extraction method. First, term level evaluation assesses the performance of extracting domain relevant terms from the corpus. Second, an extraction quality evaluation stage assesses the quality of the extracted entities.

To measure the performance of information extraction system, usually consider two performance index, precision and recall, that is define as following [27,16]:

For one web document:

Definition:

1. **Information entity**: An IT product specification that may be formed be one or several lexical token.
2. **Target information entity**: An IT product specification that meets our information extraction scope and hide in selected web document
3. **Information entity extracted**: A lexical construction that is extracted by JAPE grammar rule.
4. **Correct information entity extracted**: A correct IT product specification that is extracted by JAPE grammar rule and.

Base on above definition, the recall and precision rate be defined as:

$$\textbf{Recall} = \frac{\textbf{Number of correct information entities extracted}}{\textbf{Number of target information entities}}$$

$$\textbf{Precision} = \frac{\textbf{Number of correct information entities extracted}}{\textbf{Number of information entities extracted}}$$

We use the following step to evaluate the performance of IT product specification extraction system [28].

Step 1. Manually annotation:

We annotate all selected web documents manually to mark up all target information entities. It is baseline of performance evaluate.

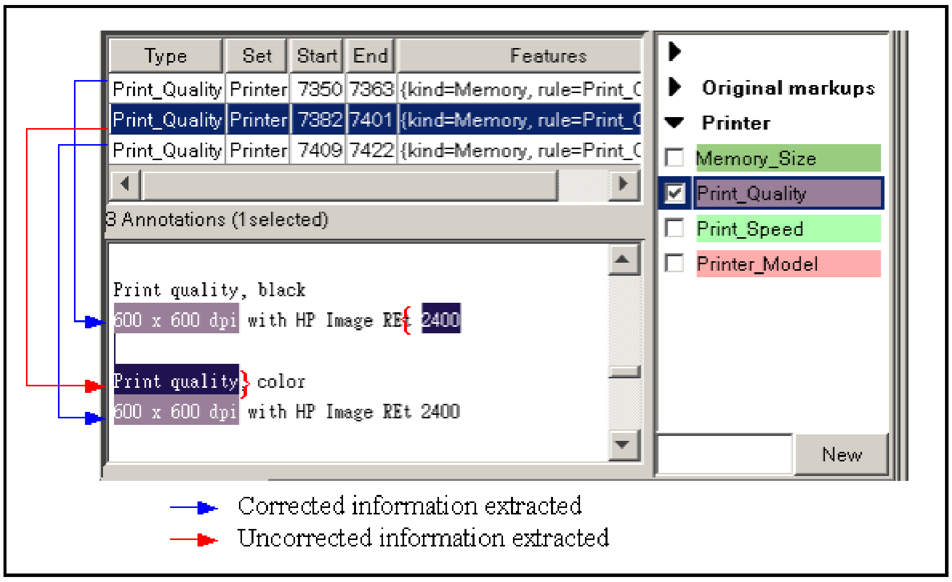
Step 2. Automatic annotation by JAPE grammar rule:

We run all application of IT product specification extraction system, to mark up and extract information entities from selected web page by automatic process.

Step 3. Compare and calculate the recall and precision:

Compare the information entities that mark up by manual and extract by automatic process to calculate the recall and precision values.

Figure 25 is an example to explain the calculation of recall and precision. It is a web page segment that describes the print quality of printer. Annotate the text segment by manually, we know the black and color printer quality both are 600x600 dpi, so there are 2 target information entities show be extracted. But JAPE grammar rule extracts 3 information entities, two are correct and one is missed.



Type	Set	Start	End	Features
Print_Quality	Printer	7350	7363	{kind=Memory, rule=Print_C
Print_Quality	Printer	7382	7401	{kind=Memory, rule=Print_C
Print_Quality	Printer	7409	7422	{kind=Memory, rule=Print_C

3 Annotations (1 selected)

```
Print quality, black
600 x 600 dpi with HP Image REt 2400

Print quality color
600 x 600 dpi with HP Image REt 2400
```

Figure 25: Example of recall and precision rate calculation

So we get following result:

Target information entity = 2

Information entity extracted = 3

Correct information entity extracted = 2

Recall = 2 / 2 = 100%

Precision = 2 / 3 = 66.6%

We run all application and calculate recall and precision rate for application, the result show as following table:

Personal Computer Application

Table 9: Recall and Precision of personal computer extraction result

Product name	Number of target information entities (A)	Number of information entities extracted (B)	Number of correct information entities extracted (C)	**Recall** (C/A)	**Precision** (C/B)
HP Pavilion dv4030us Notebook	10	11	10	100%	90.9%
HP Pavilion zv6010us Notebook	10	10	10	100%	100%
HP Media Center zd8110us Notebook	8	9	8	100%	88.9%
HP Pavilion a1040n Desktop	10	12	10	100%	83.3%
HP Media Center m7070n Photosmart PC	9	10	9	100%	90%
IBM ThinkPad T42 Notebook	10	11	10	100%	90.9%
IBM ThinkPad X41Tablet	10	10	10	100%	100%
IBM ThinkPad R50e Notebook	9	10	9	100%	90%
IBM ThinkCentre S50 Desktop	9	10	9	100%	90%
IBM ThinkCentre M51 Desktop	9	10	9	100%	90%
Sum/Average	94	103	94	100%	91.3%

•

•

•

- **Unix server Application**

Table 10: Recall and Precision of Unix server extraction result

Product name	Number of target information entities (A)	Number of information entities extracted (B)	Number of correct information entities extracted (C)	**Recall** (C/A)	**Precision** (C/B)
HP RX8620	18	18	18	100%	100%
HP RX7620	16	16	16	100%	100%
HP RX4640	25	25	25	100%	100%
HP RP8420	19	16	16	84.2%	100%
HP RP7420	18	15	15	83.3%	100%
IBM p5 590	14	13	13	92.9%	100%
IBM p5 570	18	18	16	88.9%	88.9%
IBM p5 550	10	11	10	100%	90.9%
IBM p5 520	10	11	10	100%	90.9%
IBM p5 510	10	11	10	100%	90.9%
Sum/Average	158	154	149	95%	96.2%

- **Monitor Application**

Table 11: Recall and Precision of monitor product extraction result

Product name	Number of target information entities (A)	Number of information entities extracted (B)	Number of correct information entities extracted (C)	**Recall** (C/A)	**Precision** (C/B)
HP vs1717 Flat-Panel LCD Monitor	10	11	10	100%	90.9%
HP f1905 19 LCD Flat-Panel Monitor	11	11	11	100%	100%
HP Pavilion mx704 17 Flat-Screen CRT Monitor	12	11	11	91.7%	100%
HP s7540 CRT Monitor	13	11	11	84.6%	100%
IBM ThinkVision L170p Monitor	25	27	24	96%	88.9%
IBM ThinkVision L150 Monitor	20	19	19	95%	100%
IBM E74M 17 inch CRT Monitor	21	25	21	100	84%
IBM E54 15" Monitor MPRII	11	10	10	90.9%	100%
SUN/Average	123	125	117	95.1%	93.6%

-

- **Printer Application**

Table 12: Recall and Precision of personal computer extraction result

Product name	Number of target information entities (A)	Number of information entities extracted (B)	Number of correct information entities extracted (C)	**Recall** (C/A)	**Precision** (C/B)
IBM Infoprint Color 1357 Printer	6	5	5	83.3%	100%
IBM Infoprint 1412 Printer	10	9	9	90%	100%
IBM Infoprint 1352 Printer	9	8	8	88.9%	100%
HP Inkjet 1200dn printer	17	21	17	100%	81%
HP Deskjet 3845 Printer	6	6	6	100%	100%
HP Color LaserJet 2600n Printer	10	11	10	100%	90.9%
SUM/Average	58	60	55	94.8%	91.7%

The following is performance summary table of IT specification extraction system:

- **Summary Table of recall and precision:**

Table 13: Summary table of recall and precision

Product name	Number of target information entities (A)	Number of information entities extracted (B)	Number of correct information entities extracted (C)	**Recall** (C/A)	**Precision** (C/B)
Personal Computer	94	103	94	100%	91.3%
Unix Server	158	154	149	95%	96.2%
Monitor	123	125	117	95.1%	93.6%
Printer	58	60	55	94.8%	91.7%
SUM/Average	433	442	418	96.5%	94.6%

The above tables show that JAPE grammar rules have high recall and precision value to extract information entities form IBM and HP web pages.

4.3 Performance index for other company:

Since our JAPE rules are optimized for IBM and HP web pages, the high extraction performance just meets our expectation. Our major concern is the extraction performance for web pages of other company. The recall and precision values are show as following tables:

- **Personal Computer Application**

Table 14: Recall and Precision of personal computer extraction result

Product name	Number of target information entities (A)	Number of information entities extracted (B)	Number of correct information entities extracted (C)	**Recall** (C/A)	**Precision** (C/B)
Sony VGN-T370P-L Notebooks	9	10	7	78%	70.0%
Sony VGN-FS675P-H Notebook	9	8	6	67%	75.0%
Sony VAIO V167G TV-PC Desktop	14	12	10	71%	83.3%
ASUS W5A Notebook	10	6	5	50%	83.3%
ASUS W3V Notebook	9	6	5	56%	83.3%
ASUS V6V Notebook	10	6	5	50%	83.3%
Sum/Average	61	48	38	62%	79.2%

-
-
-

- **Unix server Application**

Table 15: Recall and Precision of Unix server extraction result

Product name	Number of target information entities (A)	Number of information entities extracted (B)	Number of correct information entities extracted (C)	**Recall** (C/A)	**Precision** (C/B)
Sun Fire V890 Server	13	2	2	15.4%	100%
Sun Fire V40z Server	15	10	8	53.3%	80%
Sun Fire V240 Server	13	5	5	38.5%	100%
Sun Fire V1280	16	4	4	25.0%	100%
Sun Fire E4900 Server	12	0	0	0%	--
Sun Fire E20K Server	13	0	0	0%	--
Sum/Average	82	21	19	23.2%	90.5%

- **Monitor Application**

Table 16: Recall and Precision of monitor product extraction result

Product name	Number of target information entities (A)	Number of information entities extracted (B)	Number of correct information entities extracted (C)	**Recall** (C/A)	**Precision** (C/B)
BenQ USA - FP531 LCD monitor	5	5	3	60%	60%
BenQ FP71V LCD monitor	6	5	3	50%	60%
BenQ FP71E LCD monitor	5	5	3	60%	60%
BenQ FP537s LCD Monitor	6	5	3	50%	75%
Acer AF715 CRT monitor	6	4	3	50%	75%
Acer AC501 CRT monitor	7	4	3	43%	75%
Average	35	28	18	51%	64.3%

- **Printer Application**

Table 17: Recall and Precision of personal computer extraction result

Product name	Number of target information entities (A)	Number of information entities extracted (B)	Number of correct information entities extracted (C)	**Recall** (C/A)	**Precision** (C/B)
Epson PictureMate	3	1	1	33.3%	100%
Epson Stylus Photo R300 Printer	3	2	2	66.7%	100%
Epson Stylus C66 Printer	4	3	3	75.0%	100%
CanonPIXMA iP3000 printer	6	1	1	16.7%	100%
Canon Printers - i80 Printer	9	2	2	22.2%	100%
Canon PIXMA iP90 Printer	6	1	1	16.67%	100%
SUM/Average	31	10	10	32.3%	100%

- **Summary Table of recall and precision:**

Table 18: Summary table of recall and precision for other company

Product name	Number of target information entities (A)	Number of information entities extracted (B)	Number of correct information entities extracted (C)	**Recall** (C/A)	**Precision** (C/B)
Personal computer	61	48	38	62%	79.2%
Unix Server	82	21	19	23.2%	90.5%
Monitor	35	28	18	51%	64.3%
Printer	31	10	10	32.3%	100%
SUM/Average	209	107	85	41%	79.4%

From above tables, we see the average precision is above 79.5% but the average recall value is down to 41%. It shows that the recall obviously reduces and precision still keeps acceptable level as the extension of web site scope. This result implies that JAPE grammar rules get either correct information entity or nothing.

5. Conclusion and Contribution

Information extraction is an important process in knowledge management domain. In this paper we present an approach that combine natural language processing and ontology concept to extract information for unstructured web documents, and build up a prototype of IT product specification extraction system with simple JAPE grammar rule to test the recall and precision rate.

The major challenge of information extraction methodology is how to enhance both recall and precision values. From the extraction result, we see the JAPE grammar rule has good performance for specific knowledge domain. Both of recall and precision value are higher than 85%, even 100% for some documents.

JAPE could be treated as one kind of pattern match methodology, but provide more flexibility than traditional way. JAPE grammar rule is very suitable to extract following information entity:

1. The information entity has specific text structure, like monitor resolution mode. Generally, a monitor provide several different resolution modes, like "1024x768 @ 60Hz", "800x600 @70Hz" …and so on. It is difficult to extract this kind of information by other extraction method. But the monitor resolution mode has a common text structure as:

   {number} {x} {number} ({@ | at}) {number} {Hz}

   It is easy to identify the resolution mode accurately form web document by JAPE grammar rule.

2. The information entity is composed by several tokens, like CPU cache. CPU

cache usually present as “Level1 cache: 6MB”. It will be separated several tokens as “Level”, “1”, “cache”, “:”, “6”, “MB”. Any one is common token in web document, no special keyword in this strings, it is not easy to extract the full information entity accurately by other text mining methodology. But we can descript the CPU cache as :

{Level} {nember} {cache} {:} {number} {MB}

Only the text string fully match this grammar rule will be extracted, it substantially enhance the precision rate of information extraction.

On other hand, the JAPE grammar rule is developed to extract information entities from certain web site or knowledge domain. It is difficult to widely apply the same JAPE rule to different web sites. The recall and precision rate maybe down as the extension of web sites.

This approach not only provides an information extraction methodology, it also links with ontology concept and presents the extraction result with ontology language. It means that the extracted information entities not only a single item, it includes the relation or property. These extraction results maybe become the resource of post-process, such as semantic web agent or ontology-based knowledge management system.

The major contributions of our research are show as following:

1. Build up an automatic information extraction process and prototype to enhance the performance of IT product specification extraction. This process link NLP technology and Ontology concept to provide high extraction performance.

2. Convert information entities that hide in current html document to be DAML document that follow RDF (Resource Description Framework) specification. It is important to push Semantic Web popularization.

6. Reference

[1] Michio Kaku, "Visions: How Science will Revolutionize the Twentyfirst Century", Oxford, New York, Oxford University Press, 1998.

[2] Harith Alani, Sanghee Kim, David E. Millard, Mark J. Weal, Wendy Hall, Paul H. Lewis, Nigel R. Shadbolt, "Automatic Ontology-based knowledge Extraction and Tailored Biography Generation from the Web", IEEE 2003

[3] David Masterson, Nicholas Kushmerick, "Information extraction form MDT", ECML-2003 Workshop on Adaptive Text Extraction & Mining

[4] Khaled Khelif, Rose Dieng-Kuntz, "Ontology-base semantic annotation for biochip domain", KMOM Workshop ECAI2004, 2004

[5] Tim Berner's Lee, "Information Management: A Proposal", CERN, March 1989,May 1990

[6] "What is an Ontology?", http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[7] Adelaide, Australia. D. Mollá and B. Hutchinson, "Natural Language Processing in the Undergraduate Curriculum", Proc.Fifth Australasian Computing Education Conference (ACE2003),

[8] "An Introduction to NLP" http://www.cs.bham.ac.uk/~pxc/nlpa/2002/AI-HO-IntroNLP.html

[9] Tomek Strzalkowski, Barbara Vauthey, "Information Retrieval using Robust Natural Language Processing", ACL 1992: 104-111

[10] Thompson CA,Califf ME,Mooney R J, "Active Learning for Natural Language Parsingand Information Extraction", Proc. of the Sixteenth Intl. Machine Learning, June 1999

[11] Claire Cardie, "Empirical Methods in Information Extraction", AI Magazine, 18:4, 65-79 1997

[12] Marti Hearst, "What is Text Mining? ", October 17, 2003, SIMS,UC Berkeley http://www.sims.berkeley.edu/~hearst/text-mining.html

[13] "Office of Naval Research (ONR) Science & Technology" http://www.onr.navy.mil/sci_tech/special/technowatch/default.htm

[14] Kostoff RN, "Text Mining for Global Technology Watch", available on [13]

[15] Kostoff RN, "Information Extraction From Scientific Literature with Text Mining" , 2001. (available on [13])

[16] Raymond J. Mooney , Un Yong Nahm, "Text Mining with Information Extraction", In Proceed-. ings of the AAAI 2002 Spring Symposium on Mining Answers from Texts and. Knowledge Bases, 2002.

[17] W. Fan, L. Wallace, S. Rich, Z. Zhang, "Tapping into the Power of Text Mining",

Communications of ACM, forthcoming, 2005

[18] R. Cooley, B. Mobasher, J. Srivastava , "Web Mining: Information and Pattern Discovery on the World Wide Web", Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97), November 1997.

[19] N. Guarino, Trento, "Proc. of the 1st International Conference on Formal Ontology and Information System", Italy, Jun. 1998.

[20] S. Luke and J. Heflin, "SHOE 1.01 Proposed Specification", SHOE Project, Feb. 2002. http://www.cs.umd.edu/projects/plus/SHOE/spec1.01htm.

[21] I. Horrocks and F. van Harmelen, "Reference Description of the DAML+OIL Ontology Markup Language", draft report, 2001, http://www.daml.org/2000/12/reference.html

[22] Harith Alani, Sanghee Kim, David E. Millard, Mark J. Weal, Wendy Hall, Paul H. Lewis, Nigel R. Shadbolt, "Automatic Ontology-based knowledge Extraction and Tailored Biography Generation from the Web", IEEE 2003

[23] David Vallet, Miriam Fernández, and Pablo Castells, "An Ontology-Based Information Retrieval Model", Universidad Autónoma de Madrid

[24] GATE Home page, http:// http://gate.ac.uk/

[25] D. Maynard and H. Cunningham, "Multilingual Adaptations of a Reusable Information Extraction Tool", In Proceedings of the Demo Sessions of EACL'03, Budapest, Hungary, 2003.

[26] R. Navigli, P. Velardi, A. Cucchiarelli, and F. Neri, "Quantitative and Qualitative Evaluation of the OntoLearn Ontology Learning System", ECAI Workshop on Ontology Learning and Population, 2004

[27] Marta Sabou, Chris Wroe, Carole Goble, and Gilad Mishne, "Learning domain ontology for web services description", World Wide Web Conference (WWW 2005: 190-198), 2005

[28] Chun-Chun Tsai, "Spatial information extraction from Chinese News articles", Jun, 2003, NTU

7. Appendix

7.1 JAPE Grammar Rule for personal computer

```
Phase: PC_Spec
Input: Lookup Token
Options: control = appelt debug = false


Macro: DDR_RAM
(
 (
   {Token.string == DDR}
   (
    {Token.kind == number} |
    ({Token.string == II})
   )?
 )
)

//---------------------------------------------

Rule: Prod_model
(
   (
    {Token.string == Pavilion } |
    {Token.string == Presario } |
    (
      {Token.string == Media }
  {Token.string == Center }
    ) |
    {Token.string == ThinkPad } |
    {Token.string == ThinkCentre }
   )
   (
      {Token.kind == word }
      {Token.kind == number }
      ({Token.kind == word })?
   )

)
:PC
-->
:PC.Prod_Model = {kind = PC, rule = Prod_Model}


Rule: CPU_Model
(
  (
   {Token.string == Pentium}  |
   {Token.string == Celeron}  |
   {Token.string == Athlon }  |
   {Token.string == Athlon64} |
   {Token.string == Duron  }
  )
  ({Token.type == other})?
  (
    {Token.kind == number} |
    {Token.kind == word, Token.length == 1 }
   )
):CPU
-->
:CPU.CPU_Model = {kind = CPU, rule = CPU_Model}
```

```
Rule: CPU_Speed
Priority:10
(
  {Token.kind == number, Token.string <= 6}
  (
    {Token.string == "."}
    {Token.kind == number}
  )?
  (
    {Token.string == "G"}   |
    {Token.string == "GHz"}
  )
)
:CPU
-->
:CPU.CPU_Speed = {kind = CPU, rule = CPU_Speed}


Rule: False_CPU_Speed
Priority:20
(
  (
    {Token.kind == number, Token.string == 802}
    {Token.string == "."}
    {Token.kind == number, Token.string == 11}
    ({Token})+
  )
  (
    {Token.kind == number, Token.string <= 6}
    (
      {Token.string == "."}
      {Token.kind == number}
    )?
    (
      {Token.string == "G"}   |
      {Token.string == "GHz"}
    )
  )
)
:CPU
-->
{}


Rule: Memory_Type
(
  {Token.string == SDRAM } |
  (
    (DDR_RAM)
    ({Token.string == SDRAM})?
  )
)
:Memory
-->
:Memory.Memory_Type = {kind = Memory, rule = Memory_Type}


Rule: False_Memory_Size
Priority:20
(
  (
    (
      {Token.kind == number}
      {Token.string =~ "[xX]?"}
    ) |
    (
      {Token.string == "/" }
    )
  )
  (
    (
      {Token.kind == number, Token.string >= 256 }
      {Token.string == MB }
    ) |
    (
      {Token.kind == number, Token.string <= 6 }
      (
        {Token.string == "."}
        {Token.kind == number}
      )?

      {Token.string == GB }
    )
  )
)
:Memory
-->
{}


Rule: Memory_Size
Priority:10
(
  (
    {Token.kind == number, Token.string >= 256 }
    {Token.string == MB }
  ) |
  (
    {Token.kind == number, Token.string <= 4 }
    (
      {Token.string == "."}
      {Token.kind == number}
    )?
    {Token.string == GB }
  )
)
:Memory
-->
:Memory.Memory_Size = {kind = Memory, rule = Memory_Size}


Rule: Max_Memory_Size
Priority:15

(
  (
    {Token.string =~ "[mM]aximum?"}
    {Token.string =~ "[mM]emory?"}
  )
  (
    (
      {Token.kind == number, Token.string >= 256 }
      {Token.string == MB }
    ) |
    (
      {Token.kind == number, Token.string < 6 }
      (
        {Token.string == "."}
        {Token.kind == number}
      )?
      {Token.string == GB }
    )
  )
)
:Memory
-->
:Memory.Max_Memory_Size = {kind = Memory, rule = Max_Memory_Size}


Rule: HD_Type
(
  (
    {Token.string == Enhanced }
    {Token.string == IDE }
   )
   |
   (
     (
      {Token.string =~ "[Ss]erial?" }
      {Token.string == ATA }
     )
     |
     ({Token.string == SATA })
   )
)
:HD
-->
:HD.HD_Type = {kind = Memory, rule = HD_Type}


Rule: HD_Size
(
  {Token.kind == number, Token.string >= 20 }
  {Token.string == GB }
)
:HD
-->
:HD.HD_Size = {kind = Memory, rule = HD_Size}
```

7.2 JAPE Grammar Rule for Unix Server

```
Phase: Unix_Server_Spec
Input: Lookup Token
Options: control = appelt debug = false



Macro: DDR_RAM
(
 (
   {Token.string == DDR}
   (
     {Token.kind == number} |
     ({Token.string == II})
   )?
 )
)


//---------------------------------------------

Rule: Server_model
(
   (
     (
       (
        {Token.string == p } |
        {Token.string == P }
       )
      {Token.kind == number }
     ) |
    {Token.string == rp } |
    {Token.string == rx }
  )
  {Token.kind == number }
  (
    {Token.string == "-" }
    {Token.kind == number }
  )?
)
:Server
-->
:Server.Server_Model = {kind = Server, rule = Server_model}


Rule: CPU_Model
(
  (
   {Token.string == POWER} |
   (
     {Token.string == PA-} |
     {Token.string == PA}
   ) |
   (
     {Token.string == Itanium}
     ({Token.type == other})?
   )
  )
  ({Token.string == "-"}) ?
  {Token.kind == number}
)
:Server
-->
:Server.CPU_Model = {kind = Server, rule = CPU_Model}


Rule: CPU_Speed
```

```
(
  (
    {Token.kind == number, Token.string <= 3}
    (
      {Token.string == "."}
      {Token.kind == number}
    ) ?
    (
      {Token.string == "G"}   |
      {Token.string == "GHz"}
    )
  ) |
  (
    {Token.kind == number, Token.string >= 800}
    (
      {Token.string == "M"}   |
      {Token.string == "MHz"}
    )
  )
)
:CPU
-->
:CPU.CPU_Speed = {kind = CPU, rule = CPU_Speed}


Rule: CPU_Cache1
(
  (
    {Token.string =~ "[Ll]evel?" } |
    {Token.string =~ "[Ll]?" }
  )
  {Token.kind == number}
  (
     {Token.string == "("}
     {Token.kind == word}
     {Token.kind == number}
     {Token.string == ")"}
  )?
  {Token.string =~ "[Cc]ache?" }
  (
     {Token.string == "("}
     {Token.kind == word}
     {Token.string == ")"}
     {Token.kind == punctuation}
  ) ?
  (
    {Token.kind == punctuation}|
    {Token.kind == word} |
    {Token.kind == number}
  ) ?

  {Token.kind == number}
  (
    {Token.string == "."}
    {Token.kind == number}
  ) ?
  (
    {Token.string == "KB"}   |
    {Token.string == "MB"}
  )
)
:CPU
-->
:CPU.CPU_Cache = {kind = CPU, rule = CPU_Cache1}


Rule: CPU_Cache2
(
  {Token.kind == number}
  (
    {Token.string == "."}
    {Token.kind == number}
  ) ?
  (
    {Token.string == "KB"}   |
    {Token.string == "MB"}
  )
  (
    {Token.string == "shared"} |
    {Token.string == "combined"}
  ) ?
  (
    {Token.string =~ "[Ll]evel?" } |
    {Token.string =~ "[Ll]?" }
  )
  {Token.kind == number}
  {Token.string =~ "[Cc]ache?" }
)
:CPU
-->
:CPU.CPU_Cache = {kind = CPU, rule = CPU_Cache2}



Rule: Memory_type
(
  {Token.string == SDRAM } |
  (
    (DDR_RAM)
    ({Token.string == SDRAM} )?
  )
  (
    {Token.kind == number }
    {Token.string == MHz }
  )
)
:Memory
-->
:Memory.Memory_type = {kind = Memory, rule = Memory_type}


Rule: False_Memory_Size
Priority:20
(
  {Token.string == "."}
  {Token.kind == number }
  (
    {Token.string == MB } |
    {Token.string == GB }
  )
)
:Server
-->
{}

Rule: Memory_Size1
Priority:10
(
  {Token.kind == number }
  (
    {Token.string == GB } |
    {Token.string == MB }
  )
  (
    {Token.string == to }
    {Token.kind == number }
    (
      {Token.string == GB } |
      {Token.string == MB }
    )
  )
  (
    {Token.string == of }
    {Token.kind == number }
    {Token.string == MHz }
  )?
)
:Server
-->
:Server.Memory_Size = {kind = Memory, rule = Memory_Size1}


Rule: Max_Memory_Size1
Priority:50
(
  {Token.kind == number }
  {Token.string == GB }
  (
    {Token.string =~ "[Mm]ax?" } |
    {Token.string =~ "[Mm]aximum?" }
  )
)
:Server
-->
:Server.Max_Memory_Size = {kind = Memory, rule = Max_Memory_Size}


Rule: Max_Memory_Size2
Priority:45
(
  (
    {Token.string =~ "[Uu]p?"  }
    {Token.string == to }
  )
  {Token.kind == number }
  (
    {Token.string == GB } |
    {Token.string == TB }
  )
  (
    {Token.kind == number }
    {Token.string == MHz }
  )
)
:Server
-->
:Server.Max_Memory_Size = {kind = Memory, rule = Max_Memory_Size2}


Rule: OS_ver
(
  (
    {Token.string == AIX } |
    {Token.string == HP-UX }
  )
  {Token.kind == number }
  (
    {Token.string == "."}
    {Token.kind == number }
  ) ?
  ({Token.kind == word}) ?
  (
    {Token.string =~ "[Vv]ersions?" } |
    {Token.string =~ "[Vv]?" }
  )?
  (
    {Token.kind == number }
    (
      {Token.string == "."}
      {Token.kind == number }
    ) ?
    (
      {Token.string == "/"}
      {Token.kind == number }
      {Token.string == "."}
      {Token.kind == number }
    )*
  )
)
:Server
-->
:Server.OS_Ver = {kind = OS_Ver, rule = OS_Ver}


Rule: HD_Type
(
  (
    {Token.string == Enhanced }
    {Token.string == IDE }
   )|
   (
     (
      {Token.string =~ "[Ss]erial?" }
      {Token.string == ATA }
     )
     |
     ({Token.string == SATA })
   ) |
   (
     (
       {Token.string == Ultra }
       {Token.kind == number }
     )
       {Token.string == SCSI }
   )
)
:Server
-->
:Server.HD_Type = {kind = HD_Type, rule = HD_Type}
```

7.3 JAPE Grammar Rule for Monitor

```
Phase: Monitor_Spec

Input: Lookup Token
Options: control = appelt debug = false
//SpaceToken


Macro: Monitor_Brand
(
 (
   {Token.string == ThinkVision  }  |
   {Token.string == Pavilion}  |
   {Token.string == Presario }  |
   {Token.string == Lenovo } |
   {Token.string == HP }
 )
)


//----------------------------------------------

Rule: Monitor_model
(

  (Monitor_Brand)
  ({Token.kind == punctuation}) ?
  ( {Token.kind == word } ) ?
  ( {Token.kind == word } ) ?
  {Token.kind == word }
  {Token.kind == number }
  (
    {Token.kind == word, Token.length == 1 }
  )?
)
:Monitor
-->
:Monitor.Monitor_Model = {kind = Monitor, rule = Monitor_Model}


Rule: Monitor_type
(
   (
     {Token.string == TFT } |
     {Token.string == LCD } |
     {Token.string == CRT } |
     {Token.string =~ "[Pp]anel?"}
   )
   (
     {Token.string == Monitor } |
     {Token.string == monitor }
   )
)
:Monitor
-->
:Monitor.Monitor_type = {kind = Monitor, rule = Monitor_type}


Rule: Monitor_Size1
(
  {Token.kind == number, Token.string >= 14 , Token.string <= 20}
  (
   {Token.kind == punctuation}
   {Token.kind == number}
  ) ?
  (
    {Token.kind == punctuation}
  ) ?
  (
    {Token.string == "\""} |
    {Token.string == inch }
  )
)
:Monitor
-->
:Monitor.Monitor_Size = {kind = Monitor, rule = Monitor_Size1}


Rule: Monitor_Size2
(
  {Token.string  =~ "[Ss]ize?" }
  {Token.kind == number, Token.string >= 14 }
  (
   {Token.kind == punctuation}
   {Token.kind == number}
  ) ?

  {Token.kind == punctuation}
)
:Monitor
-->
:Monitor.Monitor_Size = {kind = Monitor, rule = Monitor_Size2}


Rule: False_Monitor_Size1
Priority:50
(
  {Token.kind == number, Token.string >= 14 , Token.string <= 20}
  (
   {Token.kind == punctuation}
   {Token.kind == number}
  ) ?
  (
    {Token.kind == punctuation}
  ) ?
  (
    {Token.string == "\""} |
    {Token.string == inch }
  )
  {Token.string =~ "[Xx]?" }
)
:Monitor
-->
{}

Rule: Monitor_Price
(
  {Token.string =~ "[Pp]rice?" }
  {Token.string == "$" }
  {Token.kind == number, Token.string >= 100}
  (
    {Token.string == "."}
    {Token.kind == number}
  ) ?
)
:Monitor
-->
:Monitor.Monitor_Price = {kind = Monitor, rule = Monitor_Price}


Rule: Pixel_Pitch
Priority:10
(
  {Token.string =~ "[Dd]ot?" }
  {Token.string =~ "[Pp]itch?" }
  ( {Token.string == 0 } ) ?
  {Token.string == "." }
  {Token.kind == number, Token.string <= 300}
  {Token.string == mm }
)
:Monitor
-->
:Monitor.Pixel_Pitch = {kind = Monitor, rule = Pixel_Pitch}

Rule: Not_Pixel_Pitch
Priority:20
(
  {Token.kind == number, Token.string > 0 }
  {Token.string == "." }
  {Token.kind == number, Token.string <= 300}
  {Token.string == mm }
)
:Monitor
-->
{}


Rule: Resolution_Mode
(
  {Token.kind == number, Token.string >= 300 }
  {Token.string == "x" }
  {Token.kind == number, Token.string >= 300 }
  (
    (
      {Token.string == "@" } |
      {Token.string == "at" }
    ) ?

    (
      {Token.kind == number}
      {Token.kind == punctuation}
    )*
    {Token.kind == number}
    {Token.string == "Hz" }
  )?
)
:Monitor
-->
:Monitor.Resolution_Mode = {kind = Monitor, rule = Resolution_Mode}


Rule: Max_Resolution_Mode
(
  {Token.string  =~ "[Mm]aximum?" }
  {Token.string  =~ "[Rr]esolution?" }
  {Token.kind == number, Token.string >= 300 }
  {Token.string == "x" }
  {Token.kind == number, Token.string >= 300 }
  (  {Token.string == "x" } ) ?
  (
    (
      {Token.string == "@" } |
      {Token.string == "at" }
    )
    (
      {Token.kind == number}
      {Token.kind == punctuation}
    )*
    {Token.kind == number}
    {Token.string == "Hz" }
  )?
)
:Monitor
-->
:Monitor.Max_Resolution_Mode = {kind = Monitor, rule = Max_Resolution_Mode}


Rule: Recommended_Resolution_Mode
(
    {Token.string  =~ "[Rr]ecommended?" }
    {Token.string  =~ "[Rr]esolution?" }
    {Token.kind == number, Token.string >= 300 }
    {Token.string == "x" }
    {Token.kind == number, Token.string >= 300 }
    (
      (
       {Token.string == "@" } |
       {Token.string == "at" }
      )
      (
       {Token.kind == number}
       {Token.kind == punctuation}
      )*
      {Token.kind == number}
      {Token.string == "Hz" }
    )?
  )
  :Monitor
  -->
  :Monitor.Recommended_Resolution_Mode = {kind = Monitor, rule = Recommended_Resolution_Mode}



  Rule: Refresh_Rate
  (
    (
      {Token.string == "Maximum" } |
      {Token.string == "Minimum" } |
      {Token.string == "Slowest" }
    ) ?
    (
      {Token.string == "Vertical" } |
      {Token.string == "Horizontal" }
    )
    (
      (
        {Token.string == "Refresh" }
        {Token.string == "Rate" }
      ) |
      (
        {Token.string == "Frequency" }
        {Token.string == "Range" }
      )
    )

    ({Token.string == ">" }) ?
    {Token.kind == number}
    (
      (
       {Token.string == "-" } |
       {Token.string == "to" }
      )
      {Token.kind == number}
    ) ?
    (
      {Token.string == "Hz" } |
      {Token.string == "kHz" } |
      {Token.string == "KHz" }
    )
  )
  :Monitor
  -->
:Monitor.Refresh_Rate = {kind = Monitor, rule = Refresh_Rate}
```

7.4 JAPE Grammar Rule for Printer

```
Phase: Printer_Spec

Input: Lookup Token
Options: control = appelt debug = false
//SpaceToken



Rule: Printer_model
(
  (
    {Token.string == "IBM" } |
    {Token.string == "HP" }
  )
  ({Token})?
  ({Token})?
  (
   {Token.string =~ "[Ii]nfoprint?" } |
   {Token.string =~ "[Ii]nkjet?" } |
   {Token.string =~ "[Dd]eskjet?" } |
   {Token.string == "LaserJet" }
  )
  (
   {Token.string =~ "[Cc]olor?" }
  )?
  {Token.kind == number }
  (
   {Token.kind == word, Token.length <= 2}
  )?

)
:Printer
-->
:Printer.Printer_Model = {kind = Monitor, rule = Printer_Model}



Rule: Print_Quality
(
  {Token.kind == number }
  (
   {Token.string =~ "[Xx]?" }
   {Token.kind == number }
  ) ?
  (
    (
     ({Token}) ?
     ({Token}) ?
     {Token.string =~ "[Dd]pi?" }
    ) |
    (
      ({Token})?
      {Token.string =~ "[Qq]uality?" }
    )
  )
)
:Printer
-->
:Printer.Print_Quality = {kind = Memory, rule = Print_Quality}


Rule: Print_Speed1
(
  (
   {Token.string =~ "[Cc]olor?" } |
   {Token.string =~ "[Bb]lack?" }
  )?
  {Token.string =~ "[Pp]rint?" }
  {Token.string =~ "[Ss]peed?" }
  ({Token})?
  ({Token})?
  ({Token})?
  ({Token})?
  ({Token})?
  ({Token})?
  ({Token})?
  ({Token})?
  {Token.string =~ "[Uu]p?" }
  {Token.string =~ "[Tt]o?" }
  {Token.kind == number }
  (
     {Token.string == "ppm" } |
     {Token.string == "PPM" }
  )
)
:Printer
-->
:Printer.Print_Speed = {kind = Memory, rule = Print_Speed1}


Rule: Print_Speed2
(
  {Token.string =~ "[Pp]rint?" }
  (
    {Token.string =~ "[Uu]p?" }
    {Token.string =~ "[Tt]o?" }
  ) ?
  {Token.kind == number }
  {Token.string == pages}
  {Token.string == per}
  {Token.string == minute}

)
:Printer
-->
:Printer.Print_Speed = {kind = Memory, rule = Print_Speed2}


Rule: Print_Speed3
(
  {Token.string =~ "[Mm]aximum?" }
  {Token.string =~ "[Ss]peed?" }
  ({Token})?
  ({Token})?
  ({Token})?
  ({Token})?
  ({Token})?
  ({Token})?
  ({Token})?
  {Token.kind == number }
  (
     {Token.string == "ppm" } |
     {Token.string == "PPM" }
  )

)
:Printer
-->
:Printer.Print_Speed = {kind = Memory, rule = Print_Speed3}


Rule: Memory_Size
(
  {Token.string =~ "[Mm]emory?" }
  ({Token}) ?
  ({Token}) ?
  ({Token}) ?
  ({Token}) ?
```

```
    ({Token}) ?
    ({Token}) ?
    ({Token.string =~ "[Bb]ase?" }) ?
   {Token.kind == number }
    (
      ({Token}) ?
     {Token.kind == number }
   )?
   {Token.string == "MB" }

  )
  :Printer
  -->
  :Printer.Memory_Size = {kind = Memory, rule = Memory_Size}
```