IMPROVING PRODUCT-RELATED PATENT INFORMATION ACCESS WITH AUTOMATED TECHNOLOGY ONTOLOGY EXTRACTION

WANG JINGJING

(*B. Eng.*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2013

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Wang Jing Jing
8 Jan 2013

WANG JINGJING
8 January 2013

ACKNOWLEDGEMENTS

Firstly, I am grateful to my supervisors Prof. Lu Wen Feng and Prof. Loh Han Tong, for their supervision and help. I would like to thank Prof. Fuh Ying Hsi the examiner of my PhD written Qualifying Examination. Moreover, I would like to thank panel members of my PhD oral Qualifying Examination, also examiners of my thesis and oral defense: Prof. Poh Kim Leng and Prof. Ang Marcelo Jr Huibonhoa. I would also like to thank Prof. Seah Kar Heng, the chairman of my oral defense.

Next, I would like to thank my seniors - Prof. Liu Ying and Dr. Zhan Jiaming. I appreciate their suggestions and help. I also want to thank Prof. Fu Ming Wang for his kindness, help and encouragement.

Then, I want to thank my friends, including Dr. Gong Tianxia (Centre for Information Mining and Extraction, NUS); Dr. Xue Yinxing (Data Storage Institute, A*STAR); Dr. Liu Xin, and Mr. Tu Weimin (Bioinformatics and Drug Design group, NUS); Dr. Mu Yadong (Digital Video Multimedia Lab, Columbia University); Dr. Yan Feng (Harvard University); and finally Dr. Niu Sihong, Dr. Fang Hongchao and Dr. Li Haiyan (manufacturing division, Department of Mechanical Engineering, NUS).

Lastly, I wish to thank my parents for their support and love.

TABLE OF CONTENTS

SUMMARY

This thesis focuses on patent text mining and knowledge reuse for product design and development. With the increase in the number of issued patents and the enhancement of patent awareness, patent disputes become more and more frequent. To facilitate information reuse and avoid patent infringement, this thesis defines a new ontology, called technology ontology and proposes a framework to utilize the technology ontology. The technology ontology emphasizes on two aspects of a technology: its effect and its structure. Two challenges were addressed: technology ontology extraction and technology comparison.

The automated model extraction was treated as a Named Entity Recognition problem and a parsing problem, respectively. The Named Entity Recognition system was recognized in a cutting edge patent information access evaluation. To realize patent claim parsing, a new dependency grammar framework was proposed. It makes efficient and effective claim parsing possible.

For the technology comparison, a new graph similarity measure is proposed. The proposed similarity measure can overcome the weakness of previous graph similarity measures. Moreover, it demonstrates its superiority in a patent classification problem.

Two applications are given. The first application is an effect-oriented patent search engine, which offers more focused search results than conventional patent search engine. The second application is a patent visualization tool attached to the effect-oriented patent search engine. It is able to automatically generate patent growth map that groups technologies and facilitates the selection of core technologies.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACE	Automatic Content Extraction
AI	Artificial Intelligence
ASTN	Average Similarity of True Negative
BIO	Begin, Inside, Outside
CAD	Computer-Aided Design
CRFs	Conditional Random Fields
DIPRE	Dual Iterative Pattern Relation Extraction
E-model	Effect model
EPO	European Patent Office
E-S model	Effect-Structure model
HMM	Hidden Markov Model
HTML	HyperText Markup Language
ICSDS	Independent Claim Segment Dependency Syntax
IE	Information Extraction
IP	Intellectual Property
IPC	International Patent Classification
IR	Information Retrieval
JPO	Japan Patent Office
KIPO	Korean Intellectual Property Office
k-NN	*k*-Nearest Neighbor
MPEP	Manual of Patent Examining Procedure
MUC	Message Understanding Conference
MuST	Multi-modal Summarization for Trend task

NCPA	New Comprehensive Patent Analysis approach
NER	Named Entity Recognition
NII	National Institute of Informatics
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
NTCIR	NII Test Collection for IR systems
OIE	Open Information Extraction
OWL	Web Ontology Language
PMO	Patent Metadata Ontology
PGM	Patent Growth Map
POS	Part-Of-Speech
RE	Relation Extraction
SAO	Subject-Action-Object
SIPO	State Intellectual Property Office of the People's Republic of China
S-model	Structure model
SUMO	Suggested Upper Merged Ontology
SVM	Support Vector Machine
TCMLs	Traditional Chinese Medicine Language system
TRIZ	TIPS, the Theory of Inventive Problem Solving
UML	Unified Modeling Language
USPC	U.S. Patent Classification system
USPTO	United States Patent and Trademark Office
VSM	Vector Space Model
WIPO	World Intellectual Property Organization

CHAPTER 1

INTRODUCTION

1.1 Background

A patent is an official document, and a form of Intellectual Property (IP). A patent also refers to a right. The United States Patent and Trademark Office (USPTO) defines a patent as an IP right granted by the Government of the United States of America to an investor "to exclude others from making, using, offering for sale, or selling the invention throughout the United States or importing the invention into the United States" for a limited time in exchange for public disclosure of the invention when the patent is granted. This right has been established over 200 years. The first United States Patent Act was passed into law in 1790. The United States Constitution, which was adopted in 1789, is the foundation of the patent law.

A product-related patent refers to any patent that contains information pertaining to product design and development. Such information includes but is not limited to a product, a design, a technology, a process or a kind of material. From an engineering angle, a product must be engineered, discrete, and physical (Ulrich & Eppinger, 2008). This definition excludes magazine, sweater, or software from the scope of the product.

Product-related patents are important for avoidance of IP dispute and breakthrough of technical barriers. With the increase in the number of issued patents and the enhancement of people's patent awareness, patent disputes become more and more frequent. A recent example is about Google, Microsoft and Apple. David Drummond, the senior vice president and chief legal officer of Google, released a blog entitled "when patents attack Android" on 3 August 2011. David said that Android's success has yielded something else: a hostile organized campaign against Android by Microsoft, Oracle, Apple and other companies, waged through bogus patents; they are doing this by seeking $15 licensing fees for every Android device and attempting to make it more expensive. David pointed

out that a smart phone might involve as many as 250,000 (largely questionable) patent claims, and the competitors want to impose a "tax" for these dubious patents that makes Android devices more expensive for consumers. On 22 May 2012, Google acquired mobile phone maker Motorola Mobility. This deal was worth $12.5 billion. Google said its purchase is based in large part on Motorola Mobility's large stash of patents.

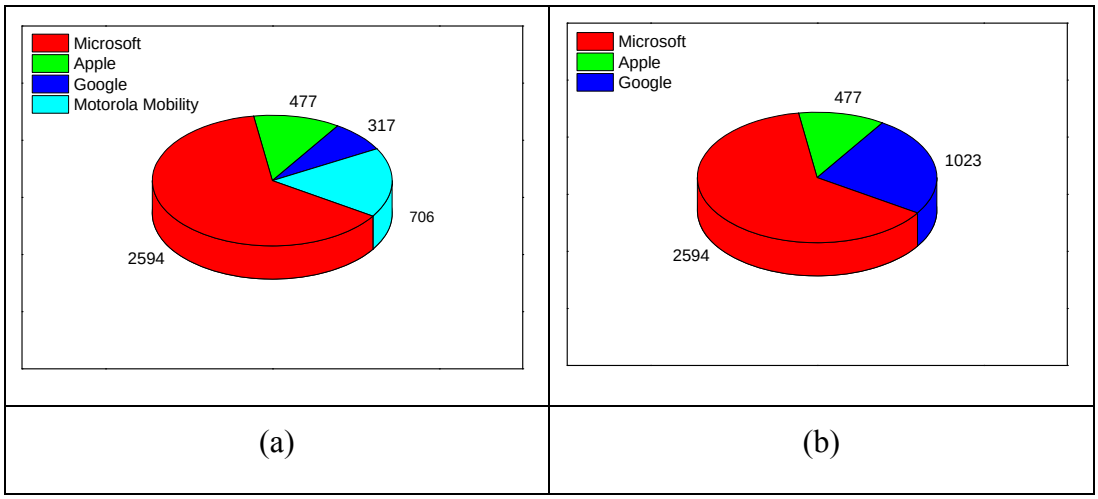


Figure 1-1 The share change based on the number of patents related to mobile device

According to data from MDB Capital, which is Wall Street's only IP investment bank, Google only had 317 patents related to mobile device at the beginning of August 2011. In contrast, the number of patents related to mobile device owned by Microsoft and Apple is 2594 and 477, respectively. It means that Google, compared to its two major competitors, is in the worst position, as shown in Figure 1-1(a). The acquisition of Motorola Mobile gives Google a total of 1023 mobile device patents, tripling Google's store of patents and overtaking that of Apple, as shown in Figure 1-1(b). The acquisition helps Google to maintain its growth in the mobile device industry. That may be why it was reported that if Google successfully acquires Motorola Mobility, a new era of IT troika will dawn.

The value of patents is not limited to IP right; patents are important available source of knowledge that can support technology reuse and facilitate product design and development. Patents provide lots of novel and complete ideas, which usually cannot be found in other publications. As an exchange of IP right, a patent must disclose complete and detailed information about how to make the invention and how to use the invention, by which anyone in the same industry can easily

understand, use and make the invention. Patent databases are often more effective for innovative requirements gathering than academic publications and thesis databases (Engler & Kusiak, 2008).

Therefore, the importance of patent search step in the product design and development process (Ulrich & Eppinger, 2008) should be highlighted. In practice, the efficiency and effectiveness of patent search and analysis relies on available patent processing tools.

1.2 Motivations

This study is motivated by the weakness of current patent search and patent analysis methodologies and the progress of two product-related text information extraction problems: relational model extraction and functional model extraction.

1.2.1 Current Patent Information Access

Current patent information access means, including patent search engines and patent analysis tools, are designed for general use. They are usually too general and may not support product design and development well. Thirty different implementations of patent management systems were studied (Briggs, Iyer & Carlile, 2007) and it was concluded that current technologies are typically used by individuals with a general understanding, such as consultants or academics, and are less useful for technical specialists or attorneys that require detailed knowledge about specific technical domains.

Patent search engines are designed for searching and querying. Anyone of the World's five major patent offices, namely United States Patent and Trademark Office (USPTO), European Patent Office (EPO), Japan Patent Office (JPO), State Intellectual Property Office of the People's Republic of China (SIPO), and Korean Intellectual Property Office (KIPO), had built its own patent database and search engine. Moreover, a patent classification system is usually built to organize and manage patents, and to facilitate patent retrieval in a specific domain. Typical patent classification systems are U.S. Patent Classification (USPC) system, Japanese F-term system, and International Patent Classification (IPC) of World Intellectual Property Organization (WIPO).

Patent analysis tools are designed for abstracting and theorizing. They usually start from a set of patents that are obtained from a patent search engine. Moreover, they often offer visualization function to enhance information access. Methodologically, patent analysis relies on citation analysis (Han & Park, 2006), keyword-based document representation (Lee, Jeon & Park, 2011; Lee, Yoon & Park, 2009) and bibliometrics. The keyword-based document representation represents a document in terms of words it contains. In Vector Space Model (VSM), a patent document is typically digitalized into a vector, each entry of which corresponds to a meaningful term or theme (Manning, Raghavan & Schütze, 2008). The co-occurrence of keywords can be utilized for classification or clustering, e.g., keyword-based similarity measures for patent clustering (Yoon B. & Y. Park, 2004). In the ThemeScape map of the Thomson Reuters, peaked mounds represent a concentration of documents and their relevance to one another is determined by proximity. Bibliometrics are a set of methods to quantitatively analyze scientific and technological literature. Such quantitative patent analysis (Wberry, 1995; Hunt, Nguyen & Rodgers, 2007) is based on numerical statistics of patents' bibliographical information (or meta-data), for example, the number of patent applications, assignees, or inventors. The obtained numbers would be further ranked and visualized as a ranking map. For example, a column chart where companies are ranked in terms of the number of patents they own, as shown in Figure 1-2. The company with the largest number of patents is considered as the dominant company, although this map does not consider any technology details involved in the patents.

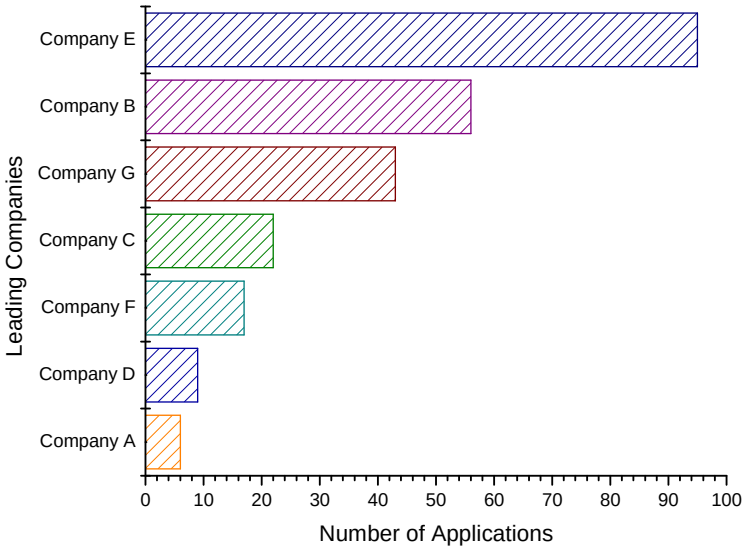


Figure 1-2 An example of ranking map

Patent search module and patent analysis module are usually integrated into a single commercial system e.g., Patsnap™ and Goldfire®, an Optimal Decision Engine. The Patsnap™ includes a search engine module and a bibliometrics module. The Goldfire® includes a search engine module and an Innovation Trend Analysis (ITA) module, which mainly includes technology analysis and citation analysis. The technology analysis is based on bibliometrics.

For technology reuse, the standard Boolean model does not handle relations well in conventional search engine. In standard Boolean model, both the documents to be searched and the query are conceived as a set of terms. With the increase of issued patents, using single keyword as query may obtain too many relevant patents. A simple strategy is to use multiple keywords instead. These keywords are treated equally in standard Boolean model. However, explicit relation among these keywords may exist. For example, given the query "wireless mouse with long battery life", a paten contains all these keywords may not be the expected return, e.g., patent numbered 'US8390249 B2', where "long" is used in "long term evolution". If quotes are used in the query, e.g., "'wireless mouse' 'long battery life'", it may filter out many relevant patents. For example, the patent numbered 'US7702369 B1' and titled "Method of increasing battery life in a wireless device" does not contain "long battery life".

For avoidance of intellectual property dispute and breakthrough of technical barriers, there are limitations in current patent analysis methods. They overlook

the content of patent claim section e.g., the knowledge for avoiding patent infringement. The citation analysis does not offer rich enough information and is difficult to catch up-to-date trends due to the time lag between citing and cited patents. The bibliometrics analysis does not care about the content of patent claim section. The keyword-based analysis usually requires experts to manually identify valuable keywords. With VSM, multiple patents may be represented by the same vectors, while they actually describe different patented technologies. Moreover, VSM overlooks the intrinsic structure of the patent claim section. The claim section is the only part examined and conferred for protection. The claim is written for claiming intellectual property right that the inventor wants to protect. It must be as general as possible to maximize the scope of protection, and simultaneously it must be specific enough to be distinguished from prior art. Other parts e.g., description or drawings are for understanding and interpreting the claims, but do not provide any protection themselves.

1.2.2 Relational Model Extraction

Relational model is a mathematical model for describing the structure of data. In database theory, the basic data structure of the relational model is the table. A row in a database table implements a tuple. Each tuple element is identified by a distinct name, called attribute. Thus, the relations in relational database refer to the various tables in the database; a relation is a set of tuples. For example, a relation (table) is given in Table 1-1. The first row in above table can be represented using a 2-tuple (student: “Tom”, score: 77). In this notation, the attribute-value pairs may appear in any order.

Table 1-1 An Example of relational model

STUDENT	SCORE
Jim	77
Tom	78

A new comprehensive patent analysis (NCPA) approach for new product design was proposed (OuYang & Weng, 2011), where the critical issues are to manually identify key technology patents, and further to manually identify the technology and the corresponding technological performance in the patents. Such information can be stored in database in the form of the relational model. Each row in the table is a 2-tuple (TechnologyName, PerformanceName), where

TechnologyName denotes technology and PerformanceName denotes performance.

The relational models are also valuable for generating patent map. Matrix map, for example, demonstrates the link between two elements and where such link can be found. An example of matrix map demonstrating the link between technology and effect is shown in Figure 1-3. The underlying 2-tuple can be defined as (TechnologyName, EffectName), where TechnologyName denotes technology and EffectName denotes effect. Similarly, the underlying 2-tuple in a matrix map can be defined as (ProblemName, SolutionName), where ProblemName denotes problem, and SolutionName denotes solution (Fujii, Iwayama & Kando, 2004), or (TechnologyName, PurposeName), where TechnologyName denotes technical item and PurposeName denotes purpose. The matrix maps are used to find main stream technical fields and to support decision making on future technology development through seeking opportunities in sparse cells within them; they are also used to predict business opportunities via comparing the research and development focus of one company with that of its major competitors (Liu and Luo 2007).

	Effect A	Effect B	Effect C
Technology a	Patent #05 Patent #02	Patent #04	
Technology b			Patent #06
Technology c		Patent #07	Patent #03 Patent #01

Figure 1-3 An example of matrix map (Technology vs. Effect)

Alternatively, relational models can be integrated with time, hence showing the trend of development. For example, a set of 2-tuples (TechnologyName, PerformanceName), in which TechnologyName denotes technology and PerformanceName may be precision, which is a response variable ranging from zero to one and is extracted from a collection of technical documents. Then, a trend map can be created as shown in Figure 1-4. This map is considered as a kind of text summarization, which was conducted as the Multi-modal Summarization for Trend (MuST) task in the NTCIR-7 (Kato & Matsushita, 2008). The NTCIR

stands for National Institute of Informatics (NII) Test Collection for Information Retrieval (IR) systems.

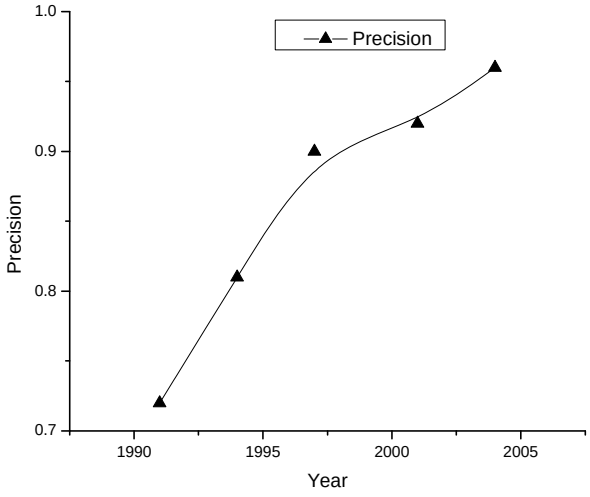


Figure 1-4 An example of technical trend map describing the changes of precision scores

1.2.3 Functional Model Extraction

A relational model is a set of tuples, while a functional model is a directed multigraph (Hung & Hsu, 2007). In such a graph, a node denotes a system or a subsystem. Different shapes can be used to differentiate different system types. An arc denotes relational action from the predecessor to the successor. More than one arc is allowed between two nodes. Both node and edge is labeled with text.

With the functional model, an integrated process for designing around existing patents was proposed (Hung Y. & Hsu Y., 2007; Yao, Jiang & Zhang et al., 2010). This method was designed for small and medium companies to develop a new product, similar to but different from an existing product, and at the same time avoiding patent infringement. The method includes four steps: searching, modeling, transforming and solving. In the searching step, a set of patents is read, and a patent is targeted. In the modeling step, the product described in the patent is modeled as a function model, and product components that can be improved are highlighted. The function model helps the designer understand the relationship (useful function, harmful function, insufficient function, etc.) between elements of the core technologies. In the transforming step, the found problems are transformed into features of TRIZ (referring to “the theory of inventive problem solving”) Contradiction Matrix, which can give some inventive principles. Those

inventive principles can inspire designers and help them to develop solutions in the final solving step. Besides, Substance-Field Analysis is used on the modified functional model following the standard TRIZ process.

The modification of the function model is shown in Figure 1-5. Briefly, Figure 1-5 (a) shows a function model; Figure 1-5 (b) highlights two components that can be improved; and Figure 1-5 (c) shows the modified function model. A detailed example can be found in (Hung & Hsu, 2007). A case study of designing spiral bevel gear milling machine was given in (Yao, Jiang & Zhang et al., 2010).

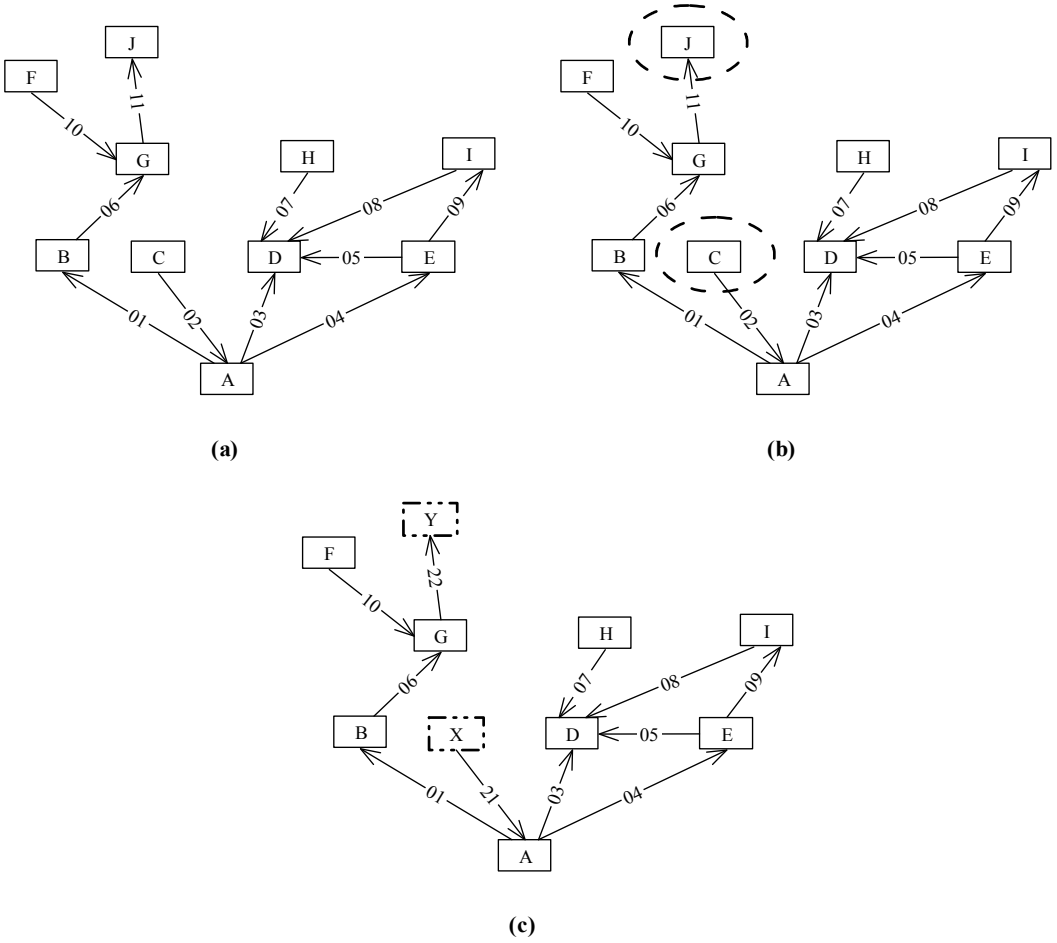


Figure 1-5 Modification process of a function model, where a rectangle denotes a component and a line denotes a function

The function model can be used for judgment of patent infringement. In general, the judgment of patent infringement consists of two principles: "all elements rule" and "doctrine of equivalents" (Hung Y. & Hsu Y., 2007). According to "all elements rule" principle, a technology infringes a patent, if all of the claim's elements of the patent are found in a technology. According to

"doctrine of equivalents" principle, if the elements in a technology corresponding to those in the claims substantially use the same way, perform the same function, and obtain the same result, then those elements is considered to be equivalent to those in the claim. A process of patent infringement avoidance is also supported by Goldfire®.

1.2.4 Specific Patent Information Access

To overcome the weakness of current methodologies and to better satisfy the requirements of product design and development, more specific information is desired. For example, relational model can be utilized to enhance technology reuse in patent search, while functional model can be utilized to consider avoidance of intellectual property dispute and breakthrough of technical barriers in patent analysis.

However, it is desirable that both relational model and functional model can be automatically extracted from text. Manual model generation requires lots of human effort, and is time consuming.

Moreover, it is desired that the technology described in a patent can be described by a model that can be automatically compared. Automated technology model comparison can facilitate analyzing and targeting key technologies, and at the same time avoiding patent infringement. Previous work (Hung & Hsu, 2007) ensures that the new design does not infringe the target patent. However, the new design may still infringe other patents. With the automated technology model comparison, avoidance of patent infringement among multiple patents can be easily achieved.

1.3 Hypothesis

This thesis is as the filler for the research gaps discussed above. The hypothesis is as follows:

(1) The product-related patent information access can be improved by better patent processing and analysis.
(2) The effectiveness is improved by utilizing additional helpful knowledge.
(3) The helpful knowledge can be represented.

(4) The efficiency is guaranteed by automatic extraction of the represented knowledge from free text.

1.4 Technology Ontology

To validate the hypothesis, the helpful knowledge is defined as technology ontology. Ontology was originally proposed (Gruber, 1993) as an explicit specification of conceptualization. The term is borrowed from philosophy, where ontology is a systematic account of existence. It should not be confused with epistemology, which is about knowledge and knowing. Ontology is further defined as a formal, explicit specification of a shared conceptualization (Studer, Benjamins, & Fensel, 1998). "Conceptualization" refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. "Shared" means the ontology is accepted by a group. "Explicit" means that the type of concepts used, and the constraints on their use are explicitly defined. "Formal" means the ontology should be machine-readable.

Briefly, ontology is a description of concepts and relationships that can exist for an agent or a community of agents. Moreover, ontology is designed for enabling knowledge sharing and knowledge reuse. Ontology is able to provide structured language and explicate the relationship between different terms; thus intelligent agent can explain flexibly its meaning without ambiguity (Uschold & Gruninger, 1996). Ontology is usually written as a set of definitions of formal vocabulary due to its nice properties for knowledge sharing among Artificial Intelligence (AI) software. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented, and the describable relationship among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge.

1.4.1 Definition of Technology Ontology

In this study, two technology-related concepts are highlighted: effect and structure. The effect is used for technology search and reuse from a teleological view, while the structure is used for technology comparison and avoidance of patent infringement in terms of claimed elements. Therefore, the Technology

Ontology primarily includes two models: an effect model (E-model) and a structure model (S-model).

An effect is defined as property changes of a patient, which is directly involved in or affected by the happening. Thus, the effect is modeled as a tuple. Typically, an effect model is a 3-tuple (or triple) denoted as (TechnologyName, PropertyName, PropertyChange), where TechnologyName denotes a technology i.e., the agent of the effect, PropertyName denotes property name and PropertyChange denotes property change. The property change can have many forms. It may be a trend, e.g., increasing in size or number, a state, e.g., temperature is 80°C, or an interval, having left and right endpoints. For example, a mouse with high battery life is modeled as a triple (Technology: "mouse", Property Name: "battery life", Property Change: "high"). This modeling method allows multiple effects to a technology.

A structure is described by all components of a technology and their relationships. Thus, the structure can be modeled as a graph. In mathematics, a graph is an abstract representation of a set of objects where some pairs of the objects are connected by links. The interconnected objects are called vertices or nodes, and the links that connect some pairs of vertices are called edges. A graph is usually depicted in diagrammatic form as a set of dots for the vertices, joined by lines or curves for the edges. In such a structure, a node denotes a technology, and an edge denotes a relation between two technologies. Typically, the structure is modeled as a tree. A tree is an acyclic connected graph where each node has zero or more children nodes and at most one parent node. In such a tree, the root node denotes the technology. Each non-root node denotes a component of a technology. A directed edge from a parent node to a children node represents the "has-part" relation.

1.4.2 Examples of S-Model Generation

The tree model is used to represent the technology's structure. The text supporting S-Model extraction can be found in the claim section of patent (Yang, Lin & Lin et al., 2005). In some patents, the structure information can also be found in the referred embodiment section. For example, the claim section of the patent numbered US6182321 is as follows:

I claim:

1. A toothbrush having an elongate handle with a longitudinal axis, a rigid curved axle extending forward generally along said longitudinal axis from one end of said handle, and a hollow integrally formed shank and toothbrush head formed of flexible plastics material that rotatable fits over said rigid curved axle along its length such that rotation of said head or shank between ±180° with respect to said curved axle causes said toothbrush head to take up different desired curved orientations.

2. A toothbrush according to claim 1, in which said axle is formed of metal.

3. A toothbrush according to claim 1, in which said shank and toothbrush head are removably fitted to said axle.

4. A toothbrush according to claim 1, in which said shank is integrally provided with peripheral finger-grippable formations.

The claim section consists of four claims. The first claim is an independent claim. The other three claims are dependent claims, which are dependent on the first claim. In the independent claim, a toothbrush is claimed and includes three components i.e., an elongate handle, a rigid curved axle and a hollow integrally formed shank and toothbrush head. The third component actually is combined with two smaller components i.e., a shank and a head. The fourth claim supplements one more component: the peripheral finger-grippable formations. The tree model of the toothbrush patented in patent numbered US6182321 is shown in Figure 1-6.

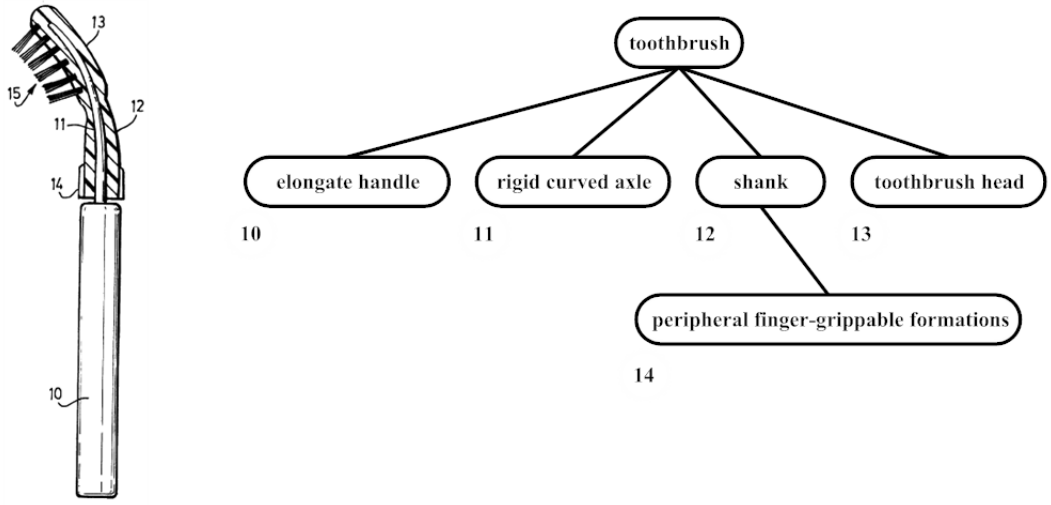


Figure 1-6 The drawing and the S-model of the patent numbered US6182321

The tree model corresponds well to the drawings of the referred embodiment, where the #10 is an elongate toothbrush handle, #11 is a stiff bent metallic wire axle, #12 is a shank, which is integrally formed with #13 i.e., a head, and #14 are finger-grippable peripheral formations. The #15 bristles are not mentioned in the

claim section, probably because they are trivia. Without #15 bristles, the tree model could still depict the patented technology well.

1.4.3 Comparison with Existent Models

The technology ontology is similar but different from the functional model. In common, both models describe a product's components. The difference is that functional model mixes functional relations and positional relations between components in the same graph, but technology ontology separates them into two models. The mixture is the deficiency of the functional model. First, two components may have multiple relations. This means multiple edges between two nodes in a graph that represents a functional model. Second, a function may be realized through multiple agents. This cannot be represented in a graph. Third, lots of relations in the functional model offer only simple position information, which is usually not considered as a very meaningful function. In contrast, the technology ontology describes structure and function (which is considered as desirable effect) separately. The S-model describes the structure of a product through its components and their positions, while an E-model can describe functions in detail and link to one or more components of the S-model.

Technology ontology is inspired by patent ontology that contains TRIZ features (Russo, 2010): the Element Name (of property) Value (of property) (ENV) model (Cavallucci & Khomenko, 2007) and Function Behavior Structure (FBS) model (Gero & Kannengiesser, 2003). Effects, similar to E-model, were collected in the scientific effects database of Goldfire®. Besides, relevance tree, similar to S-model, was adopted in normative method for technological forecasting (Martino J. P., 1993). The normative method starts with future needs and identifies the technological performance required to meet those needs. A normative forecast has implicit within it the idea that the required performance can be achieved by a reasonable extension of past technological progress (Martino J. P., 1993).

Previous works on patent ontology did not focus on implicit knowledge within patent text. Major issues covered in previous works include patent document structure, ontology language, and ontology integration. The structure of China patent was modeled as ontology (Zhi & Wang, 2009), in which a concept is a section of patent, and a relation is between two different sections. The adopted

ontology languages were Unified Modeling Language (UML) and Web Ontology Language (OWL). The ontology integration combines multiple ontologies. For European patent system, the PATExpert project (Wanner, Baeza-Yates & Brugmann et al., 2008; Giereth, Koch, & Kompatsiaris et al., 2007) defined a modular framework to integrate multiple patent ontology, including: Patent Metadata Ontology (PMO) (Gierth, Stabler & Brugmann et al. 2006), Structure Ontology, and Suggested Upper Merged Ontology (SUMO). The ontology integration can happen among different document types. For example, ontology was developed for the US patent system and integrates information in three knowledge domains: patent, court case and patent file wrapper (Taduri, Lau, & Law et al., 2011). The patent file wrapper is highly unstructured document that records prosecution history.

The knowledge contained in ontology, no matter annotated (Ghoula, Khelif, & Dieng-Kuntz, 2007) or extracted, can support many tasks, including product disassembly (Borst & Akkermans, 1997), classification (Shih & Liu, 2010), and summarization (Hwang, Miller & Rusinkiewicz, 2002).

1.5 Scope and Objectives

The scope of this thesis includes technology ontology extraction, technology comparison in terms of structure and patent information access improvement based on technology ontology.

Five objectives to be achieved are as follows:

(1) Extract automatically E-model;

It means finding effect models in the plain text of a given patent. An effect model consists of a technology as the agent of the effect, a property as the patient of the effect, and the change of the property. The specific technology, property and property's change depends on the content of the given patent.

(2) Extract automatically S-model;

It means finding the structure model with the text of the claim section of a given patent. The structure model must include a technology as a root node and at

least a component of the technology as a non-root node. The specific technology and components depend on the content of the given patent.

(3) Compare S-models;

It means measuring the difference of multiple structure models. It is used for comparing technologies.

(4) Improve patent search with E-model;

It means integrating effect model into patent search. The effect model offers additional information, and therefore can improve patent search in some aspects.

(5) Improve patent clustering with S-model;

It means integrating structure model into patent clustering. The structure models can be used for comparison of technologies and avoidance of patent infringement. The obtained additional information can enhance patent analysis.

1.6 Organization

The rest of this thesis is organized as follows: Chapter two gives a succinct literature reviews to cover major relevant research domains; Chapter three proposes a framework to summarize issues related to technology ontology and gives an introduction to all proposed methods; Chapter four proposes a method for E-model extraction; Chapter five proposes a system to utilize the extracted E-models; Chapter six gives a theoretical analysis on dependency paring of claims and proposes a new method for parsing claims; Chapter seven proposes a kind of graph similarity calculation that could be used to compare S-models; Chapter eight introduce a system that utilize S-model for patent analysis; finally, the last Chapter draws conclusions and discusses future work.

CHAPTER 2

LITERATURE REVIEW

Many currently active research domains are related to this thesis, including Text (Data) Mining, Machine Learning, Artificial Intelligence, Natural Language Processing, Information Retrieval, Statistics and Computational Linguistics. This literature review only highlights the most relevant research topics including model extraction, graph model comparison and patent map.

2.1 Ontology Learning and Ontology Extraction

Two terms are pertaining to the extraction of ontology: ontology learning and ontology extraction. Ontology learning means the acquisition of a domain model from data (Maedche & Staab, 2001). Ontology learning must consider two fundamental issues: the availability of prior knowledge and the type of input (Benz, 2007). The input types are structured data, semi-structured data and unstructured data. On the other hand, ontology extraction emphasizes that the input type for extracting ontological representations is unstructured text (Gaeta, Orciuoli & Paolozzi et al., 2011).

To reduce the human effort in ontology construction, research interest in automated method for ontology construction had risen. An automatic approach constructing ontology as thesaurus through automatic identification of keywords was proposed (Ahmad & Gillam, 2005). Another approach (Gaeta, Orciuoli & Paolozzi et al., 2011) extracts relevant ontology concepts and their relationships in terms of frequency in a knowledge base of heterogeneous text documents.

Two approaches were proposed to identify and extract part names from General Motors' archives (Bratus, Rumshisky & Khrabrov et al., 2011). The goal is to develop a robust and dynamic reasoning system functioning as a repair adviser for service technicians. The first approach is an algorithm for ontology-guided entity disambiguation. It uses existing knowledge sources, such as General Motors' parts ontology and repair manuals. The second approach extracts part names via Hidden Markov Model (HMM) with shrinkage, and models observation

dependencies in the repair notes by linear-chain Conditional Random Fields (CRFs).

The sGRAPH (Zhou, Chen & Tao, 2011) is a domain-ontology-driven automated extraction system for semantic graph. It can discover knowledge from text publications in the domain of traditional Chinese medicine. The domain ontology is the traditional Chinese medicine language system (TCMLs) including a knowledge base that contains 153,692 words and 304,114 relations. The core algorithm predicts new relation through referring existing concepts and relations.

Briefly, it must be emphasized that this thesis does not focus on the acquisition of domain ontology. For patent database, lots of work is required for constructing, updating and maintaining domain ontology, because the knowledge contained in a patent usually crosses many domains, and new concepts are emerging frequently.

2.2 Patent Map Generation

Automatic patent matrix map methods also contribute to S-model extraction. To generate the matrix map, a common strategy is to mix Text Mining (Hearst 1999; Zanasi 2005; Oluikpe, Carrillo & Harding et al., 2008) techniques with manual intervention (Tseng, Lin & Lin, 2007). Since most information (over 80%) is currently stored as text, text mining is believed to have a potential high commercial value. The general text mining techniques for generating matrix map involves: summarization (Trappey & Trappey 2008), keyword and phrase extraction, term association based on co-occurrence (Deerwester, Dumais & Furnas et al., 1990; Hofmann 1999) or based on semantics (Ide & Veronis 1998; Andreevskaia and Bergler 2006), clustering (Ward, J.H., Jr. 1963; MacQueen 1967; Dunn 1973; Bezdek 1981), clustering with semantics (Choudhary and Bhattacharyya 2002; Hotho, Staab & Stumme, 2003a; Hotho, Staab & Stumme, 2003b; Hotho, Staab & Stumme, 2003c), and cluster title generation.

Alternatively, automatic method for generating matrix maps was boosted as a feasibility study task in NTCIR-4 (Fujii, Iwayama & Kando, 2004). The organizers provided participants with the patent documents retrieved by a specific topic, and participants were requested to organize those documents in a two-dimensional matrix. In total, six topics for more than 100 relevant documents were

identified. Human experts then evaluated the submitted maps. Since the task was optional, only two participant groups (Shinmori, Okumura et al. 2004; Uchida, Mano et al. 2004) submitted their maps. One group (Shinmori, Okumura et al. 2004) focused on keyword extraction and selection, and the other group (Uchida, Mano et al. 2004) focused on clustering and cluster title generation. Both of them generated too many irrelevant titles. Moreover, the cluster titles are keywords extracted verbatim from the original patent text. Since some standard titles cannot be found in the original text directly, it is impossible to generate all correct titles.

Briefly, current patent map generation cannot be accomplished automatically. Therefore, more researches are required. For example, the analysis on claims may contribute to patent map generation (Shinmori & Okumura, 2004).

2.3 Information Extraction

Information Extraction (IE) is the research domain where text extraction methods are concentrated. The earliest IE focused on Named Entity Recognition (NER). NER seeks to locate and classify atomic elements in text into predefined categories such as the names of persons, organizations, locations, etc. The term "Named Entity" was coined at the 6th Message Understanding Conference (MUC-6) in 1995. In defining IE tasks, people noticed that it is essential to recognize information units like person names, organization names, location names, time, data, money and percentage. The number of entity types had been increased, since IE became a serious large-scale research effort (Kushmerick, Weld & Doorenbos, 1997; Appelt & Israel 1999). Two hierarchies of Named Entity types, for example, had been proposed: BBN type consists of 29 types and 64 subtypes, while Sekine's extended Named Entity hierarchy is made up of 200 subtypes.

Early entity extraction systems rely on rule-based algorithms. These rules are either manually coded or automatically learned (Kushmerick, Weld & Doorenbos, 1997; Soderland 1999; Xiao, Chua & Liu, 2003). In contrast, modern systems often resort to sequence labeling method (Sarawagi 2007). Sequence labeling is a type of pattern recognition task in machine learning (Nadeau & Sekine 2007). Supervised learning algorithms execute a decomposition of an unstructured text, and then assign a categorical label to each member of the sequence of the decomposition. Typical methods are Hidden Markov Model (HMM) (Zhou & Su

2001) and Conditional Random Fields (CRFs) (Lafferty, McCallum & Pereira, 2001; Settles 2004). It was reported that CRFs is the state-of-art method for assigning labels to token sequences (Sarawagi 2007; Sha & Pereira 2003). Compared to HMM, CRFs has many advantages. Firstly, CRFs is a conditional model, which specifies the probabilities of possible label sequences, given an observation sequence. HMM is a generative model, which assigns a joint probability to paired observation. Secondly, CRFs allows arbitrary non-independent features of the observation sequence. It is not practical to represent multiple interacting features or long-range dependencies of the observations in HMM, since the inference problem is intractable. Thirdly, in CRFs, the probability of a transition between labels can depend not only on the current observation, but also on past and future observations. In contrast, HMM must make very strict independence assumptions on the observations. Lastly, CRFs overcomes the label bias problem. It means the transitions leaving a given state compete only against each other, rather than all other transitions in the model.

Sequence labeling method does not rely on rules, which are too brittle in a noisy source. Moreover, the maintenance of sequence labeling system is easier than manual rule-based system. However, it does not mean that sequence labeling method is better than rule-based method. The curse of sequence labeling method is the overheads of training. For example, it was reported that training an HMM name recognizer is more expensive than a skilled rule writer to write a rule-based name recognizer (Appelt & Israel 1999). The HMM name recognizer cost about 800 person-hours. Preparing the training data required 20 person-hours.

There also exist hybrid systems (Rosenfeld, Feldman & Fresko et al., 2006) that attempt to obtain the benefits of both methods. Besides, the choice of features is as important as the choice of methods for a good NER system (Sang & Meulder 2003). Features were usually along three different axes: word-level, list lookup and document (Nadeau & Sekine 2007).

With the availability of recognized entities, research focus of IE shifted to Relation Extraction (RE). Generally, the task regards meaningful relations between entities from plain text. The definition is varied according to different task requirements. In the simplest form, Relation Extraction (RE) is a task of extracting relation triples from free text, e.g., extracting the triple (University:

"Stanford", Relation: "located-in", Location: "California") from text "Stanford is an American private research university located in Stanford, California".

Although it is not necessary to pre-define extractable relation types (Shinyama & Sekine, 2006), entity types and relation types are usually pre-defined. The Template Relations (Miller, Crystal & Fox et al., 1998) task in the Message Understanding Conference (MUC) are limited to organization-related relationship such as employee-of, product-of, and location-of. Seven entity types and seven relation types were defined in Automatic Content Extraction (ACE) evaluation conducted by the National Institute of Standards and Technology (NIST).

The methods for RE can be supervised, partially supervised or even unsupervised. Supervised methods may consider the RE problem as a classification problem (Bunescu & Mooney, 2005; Zhao S. & Grishman R., 2005). Partially supervised methods reduce the dependence on hand-crafted training data. For example, Dual Iterative Pattern Relation Extraction (DIPRE) (Brin, 1998) requires only a small set of labeled seed instances and enables to discover author-book pairs. SNOWBALL (Agichtein & Gravano, 2000) requires a few hand-crafted extraction patterns and enables to discover corporation-headquarters pairs. To make the tedious process of extracting large collections of facts in an unsupervised, domain-independent, and scalable manner, unsupervised relation extraction was proposed (Eichler, Hemsen & Neumann, 2008). This is feasible due to the availability of named entities and dependency. KNOWITALL (Etzioni, Cafarella & Downey et al, 2005) is able to extract hypernymy ("is-a" relationship) without hand-labeled training examples. Open Information Extraction (OIE) was proposed to extract a large set of relational tuples without requiring any human input and was implemented by TEXTRUNNER (Banko, Cafarella & Soderland, et al. 2007) with the support of dependency parsing.

An algorithm was proposed to combine the advantages of supervised IE and unsupervised IE (Mintz, Bills & Snow, et al., 2009). Besides, the adopted features (Jiang & Zhai, 2007; Zhou, Su & Zhang et al., 2005; Kambhatla, 2004) generally cross three levels: lexical, syntactic and semantic. Typical features are word, phrase, entity type, syntactic parse tree, the semantic, and dependency.

Briefly, rule-based or supervised methods require manual rules or small hand-labeled corpora of a specific domain. Both resources are scarce for E-model extraction. On the other hand partially supervised or unsupervised methods are towards domain independence and unrestricted relation type. However, they must be supported by related Natural Language Processing technologies, such as semantic database (Mintz, Bills & Snow, et al., 2009) and parsing (Shinyama & Sekine, 2006).

2.4 Claim Parsing

The S-model extraction may be realized by analyzing the parsing tree. Among various grammars, dependency grammar (Nivre, 2005) is the most suitable one for information extraction due to its two characteristics. Firstly, dependency grammar explicitly expresses word-to-word relation, thus the result of dependency parsing can easily be utilized. Other grammars usually need much more effort on post-processing to obtain word-to-word relation. Secondly, the result of dependency parsing can be obtained from phrase structure (or constituency) parsing (Marneffe, MacCartney & Manning, 2006). Since phrase structure grammars occupy a high proportion in formal grammatical systems, it means many existing natural language technologies and resources can be reused on dependency parsing.

Generally, dependency parsing is classified into two categories: grammar-based parsing or data-driven parsing. The grammar-based parsing requires grammar or rules, e.g., context-free dependency grammar. The data-driven parsing does not need grammar or rules, and the parsing decisions are made based on learned models. The learned models can be classified into graph-based models (Eisner, 1996; Wang, Lin & Schuurmans, 2007), transition-based models (Yamada & Matsumoto, 2003; Nivre & Scholz, 2004) or hybrid models (Sagae & Lavie, 2006; Nivre & McDonald, 2008; Zhang & Clark, 2008).

However, most claims seem unable to parse (Parapatics P. & Dittenbach M. 2011). Therefore, more researches are needed to investigate this issue. It should be noted that a method was proposed to parse the claim into a set of discrete elements (Lin et al., 2005). However, the S-model is a graph, rather than a list.

2.5 Graph Similarity Measures

To compare S-models, graph similarity measures can be carried out, since the S-model is modeled as a graph. Generally, graph similarity measure is a two-graph comparison problem, while the process of comparing graphs is referred as graph matching (Jouili, Tabbone & Valveny, 2010).

Different graph models use different similarity measure. The Feature Directed Acyclic Graph was proposed (Li, 2011) for Computer-Aided Design (CAD) models retrieval. A 3D model was simplified with Feature Directed Acyclic Graph and then converted into a shape distribution histogram (Osada, Funkhouser & Chazelle et al., 2002), which is a vector. The similarity of two models is therefore calculated as the distance between two vectors. For two graphs, the coupled node-edge scoring (Zager & Verghese, 2008) uses the structural similarity of local neighborhoods to derive pair-wise similarity scores for nodes and uses a linear update to generate both node and edge similarity scores. The basic idea is that a node is evaluated through its neighbor nodes and edges. The idea is inspired by a famous link analysis algorithm i.e., Hyperlink-Induced Topic Search (also known as Hubs and Authorities) (Kleinberg, 1999).

In S-model, the edge represents a Boolean "has-part" relation. Therefore, the edge similarity score does not need to be updated. Moreover, the weakness of coupled node-edge scoring is that both initial node similarity and initial edge similarity disappear after a small number of iterations. The final score is dominated by the updating process. In other words, the update equation is so dominant that human's initial intuition is killed. It is weird that two graphs are considered as analog at the beginning but they are not similar at the end in terms of the calculated similarity score.

2.6 Summary

To summarize, there exist several research gaps in literature. Firstly, previous relation extraction technologies cannot be applied on patent information access for product design and development directly. That is because rules or hand-labeled corpora for E-model are unavailable, since existing resources for IE is unsuitable

for E-model extraction, e.g., entity types like person, organization and place rarely existing in the content of a patent.

Secondly, it is desirable that claims can be correctly parsed. Thus, S-model can be extracted with parsed dependency relations.

Lastly, a more reasonable graph similarity measure is desirable for graph model comparison. The graph similarity measure should hand edge similarity appropriately and keep initial intuitive similarity judgment made by human.

CHAPTER 3

TECHNOLOGY ONTOLOGY FRAMEWORK

Technology ontology connects the knowledge space of patent database with that of the enterprise. It offers an enterprise an unprecedented capability to reuse any knowledge in the entire patent space.

To summarize issues related to technology ontology, a framework for technology ontology is given in this section. Moreover, a patent processing system that involves these issues is introduced.

3.1 Framework Overview

As shown in Figure 3-1, the core of the Technology Ontology framework is technology ontology extraction. Moreover, the framework contains four modules: patent search, patent analysis, new product development and knowledge discovery.

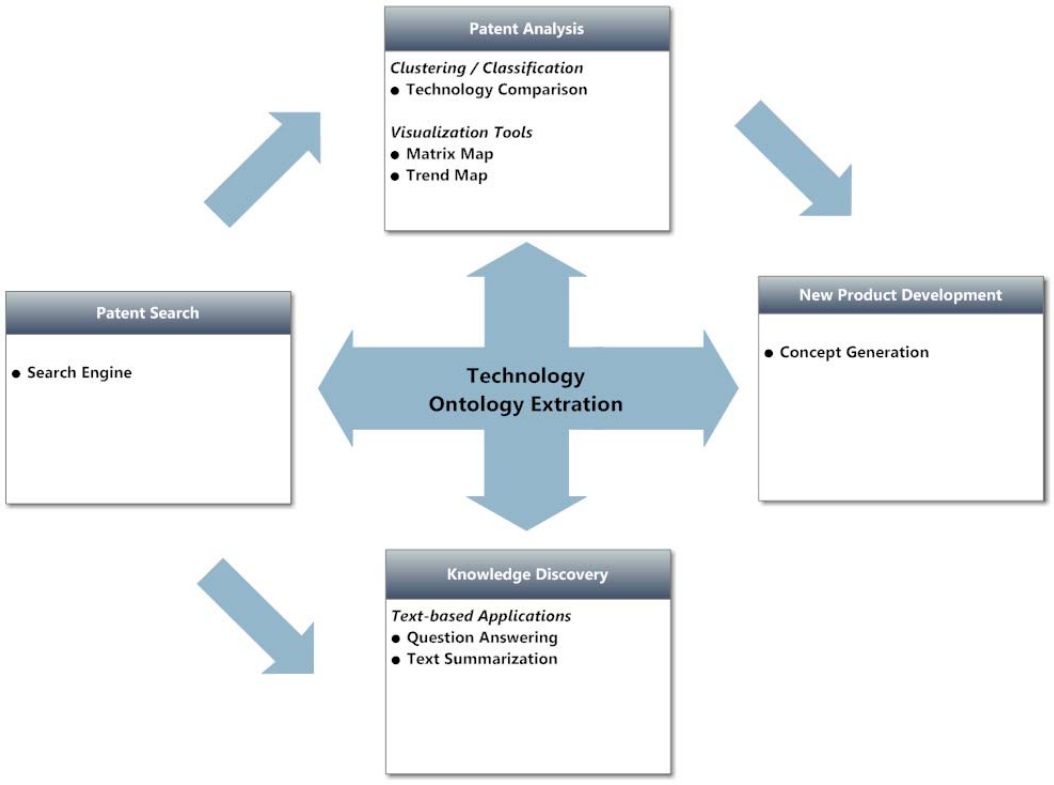


Figure 3-1 The technology ontology framework

Patent search is the Information Retrieval stage, in which a list of patent documents is retrieved. The E-model of technology ontology provides a base for

technology search and reuse from a teleological view. Product designers can search any technology that has a specific effect. A similar search manner is function-oriented knowledge search in product design and development process. Function is the base for matching customers' needs and technologies: customers' needs are identified as requirements for functions, while technologies are distinguished by their functions.

In patent analysis stage, a set of patents are analyzed and visualized. For avoidance of patent infringement, patent analysis should consider the difference of the structure. The patent technologies have similar effect, but they should be different in terms of structure. The S-model describes claimed elements of a technology in details and therefore offers a basis for technology comparison, infringement judgment, and technology selection.

In new product development stage, the S-model provides a basis for technology modification and product concept generation. A modified S-model can be easily created by changing components in an original S-model. The product design process adopting S-models can be considered as a process of disassembling and assembling, where sub-system units are selected and integrated. Therefore, the evolution of product design is the process of reselection and reintegration to satisfy the changing demand.

Besides, the obtained technology ontology can be used for other applications of knowledge discovery. Apart from facilitating relation models extraction and functional models extraction, technology ontology extraction can facilitate many text-based applications such as question answering and text summarization.

3.2 System Overview

This thesis only focuses on three modules i.e., technology ontology extraction, patent search and patent analysis in the technology ontology framework. The proposed methods can be integrated into a single patent processing system as shown in Figure 3-2.

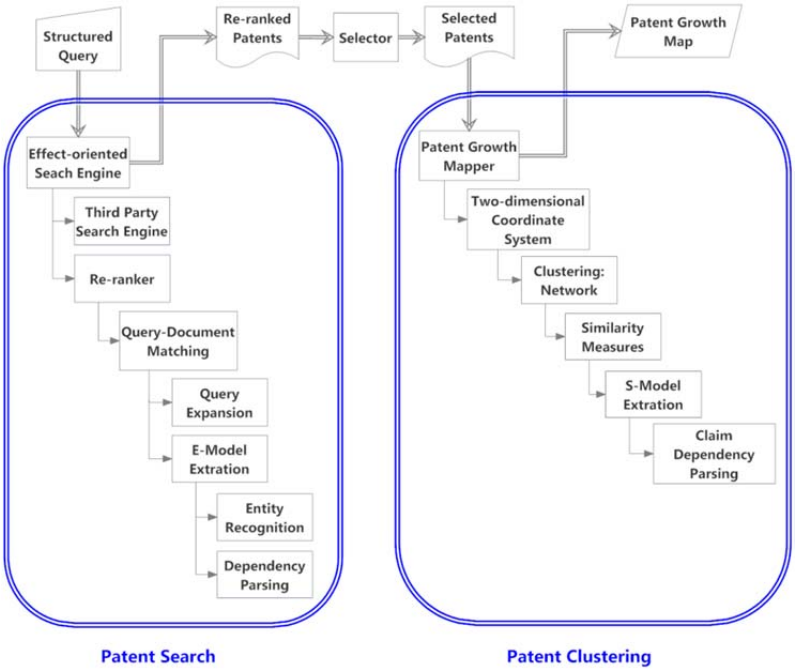


Figure 3-2 The overall system view for proposed methods

The overall system consists of two major components: an effect-oriented search engine and a patent growth mapper. The architecture of the overall system is consistent with conventional patent processing system e.g., Goldfire®, in which a patent search module is followed by a patent analysis module.

3.2.1 Effect-oriented Search Engine

The effect-oriented search engine is the patent search module. Compared to conventional patent search engine, the effect-oriented search engine involves additional effect information.

To point out the specified effect, the query of the effect-oriented search engine is structured rather than unstructured. The included effect information will affect the relevance of a patent, and affect the place of a patent on the final patent ranking.

The information integration is realized by a third party search engine and a re-ranker. The third party search engine retrieve a list of relevant patents according to the query. The re-ranker recalculates the relevance of each patent in terms of effect information the patent contains.

To know how much effect information is contained in a patent, a query-document matching that utilizes E-model is designed. Both query and document

are modeled with E-model. To enrich the natural language expression of the query, query expansion is considered to expand the single E-model given by the input query to multiple potential E-models. On the other hand, E-model extraction is carried out to model the patent document.

The E-model extraction will be considered as either an entity recognition problem or a dependency parsing problem. As an entity recognition problem, the rules or hand-labeled corpora for E-model are needed to build, since existent resources for IE are unsuitable for E-model extraction. As a dependency parsing problem, the relationship between E-model and parsing tree is needed to explore. The solution relies on the understanding of the natural language expression of E-model. Unfortunately, the natural language expression of E-model is complex, since a meaning can be expressed in many ways with natural language. Therefore, it is necessary to investigate multiple possible natural language expression manners of E-model.

3.2.2 Patent Growth Mapper

The patent growth mapper is the patent analysis module. Given a set of patents, the patent growth mapper returns a patent map, called Patent Growth Map (PGM). For avoidance of intellectual property dispute and breakthrough of technical barriers, the patent growth map utilizes S-model to cluster technologies. Technologies in the same cluster are similar in structure and are likely to infringe each other. Moreover, the patent growth map is designed with many user-friend features.

Firstly, a two-dimensional coordinate system is designed to contain a network, which is the result of technology clustering. Previous network (Yoon B. & Y. Park, 2004) did not use a coordinate system and led to arbitrary placement of dots, each of which denotes a technology or a patent. Moreover, the two-dimensional coordinate system facilitates the discovery of trend and the selection of core technology. Secondly, the number of line segments is reduced, since previous network (Yoon B. & Y. Park, 2004) uses too many line segments and is difficult to be observed. In patent growth map, the total number of line segments is controllable, while for each technology group, the number of line segments that connect dots is minimized.

To calculate the structure similarity scores used in the clustering, a more reasonable graph similarity measure is given. New graph similarity measures for S-model were proposed. They focus on node similarity rather than treating node similarity and edge similarity equally, and keep initial intuitive similarity judgment made by human.

To automatically extract S-model, claim dependency parsing is desirable so that S-model can be formed from the dependencies. However, as discussed previously, claim parsing is a challenge. To address this challenge, a new claim parsing method is proposed.

3.3 Summary

To summarize, a framework for technology ontology is given. Partial modules in the framework are highlighted and will be explored in the rest of this thesis. Moreover, major challenges and counter measures are discussed.

CHAPTER 4

EXTRACTION OF TECHNOLOGY ENTITY AND EFFECT ENTITY

The effect-oriented search engine, discussed in Chapter 3, has an E-model extraction module. In this chapter, the E-model extraction is considered as an entity recognition problem. The E-model extraction is to extract E-models from plain text of given patents. The extracted E-models are used for supporting the effect-oriented search engine. Since NER or RE researches had never focused on technology or product, resource related to E-model extraction is inadequate. Fortunately, a systematic evaluation focusing on extracting technology and effect entities was organized in NTCIR-8 (Wang, Loh & Lu, 2010). The author (Wang, Loh & Lu, 2010) had tackled this task and built a system that was ranked as the number one in terms of F-measure (Manning, Raghavan & Schütze, 2008). This chapter introduces the NER method in (Wang, Loh & Lu, 2010).

4.1 Problem Definition

The purpose of the Technical Trend Map Creation task in NTCIR-8 is to extract expressions of technologies and their effects from research papers and patents. Given the title and abstract of patents (and papers), an entity recognition system is required to label any technology entity and effect entity within the title and the abstract. Technology entity is described as algorithms, tools, materials, or data used in each study or invention. Effect entity includes one or more pairs of attribute entity and value entity. For example, effects that are expressed by a pair of an attribute and a value are shown as follows:

{[*reduce*]VALUE [*the manpower*]ATTRIBUTE}EFFECT
{[*33%*]VALUE [*redundancy-rate*]ATTRIBUTE}EFFECT

Syntactically, a “technology” or “attribute” is usually a noun or noun phrase, and a “value” can be a verb, gerund, adjective or a number.

In E-model extraction, an E-model is in the form of a triple (Technology, PropertyName, PropertyChange). For example, the triple (Technology:

"automation", PropertyName: "manpower", PropertyChange: "reduces") is required to be extracted from the text as follows:

The automation reduces manpower.

The Technical Trend Map Creation task can support E-model extraction if a one-to-one correspondence (bijection) is built: Technology Entity → Technology, Attribute Entity → PropertyName, Value Entity → PropertyChange. Thus, the triple above can be easily extracted from the labeled text as follows:

The <Technology>*automation*</Technology> <value>*reduces*</value> <attribute>*manpower*</attribute>.

4.2 Proposed Method

The task was considered as a NER problem, and a hybrid method was proposed (Wang, Loh & Lu, 2010). The hybrid method consists of a CRFs-based method and a pattern-based method.

4.2.1 Pre-processing

The pre-processing includes sentence segmentation, tokenization, POS tagging, and labeling. The sentence segmentation segments a paragraph into sentences. The developed sentence segmentation techniques are robust to HyperText Markup Language (HTML) characters noise in patent. For example, α (character code: 03B1 in Unicodehex) is "α" in HTML, but is written as ".alpha." in patent HTML file. The two dots in ".alpha." are not periods. The existence of the dots obviously causes a problem in sentence segmentation. Therefore, the sentence segmenter deletes the dot, when it belongs to a HTML character. Moreover, the developed sentence segmentation techniques are also intelligent to handle many other language situations such as suspension points, abbreviation, paper number, decimal value. For examples, dots in "i.e.", "vs." and "7.654" are not considered as periods.

The tokenization segments a sentence into tokens i.e., words, punctuations, and labels. The Part-Of-Speech (POS) tagging is a process that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc. For example, giving query:

My dog also likes eating sausage.

The tagged text is:

My/PRP$ *dog*/NN *also*/RB *likes*/VBZ *eating*/VBG *sausage*/NN ./.

The POS tags are explained as follows:

PRP$: Possessive pronoun NN: Noun, singular or mass RB: Adverb VBZ: Verb, 3rd person singular present VBG: Verb, gerund or present participle

The Stanford POS tagger (Toutanova, Klein & Manning, 2003; Toutanova & Manning, 2000) was used for both tokenization and POS tagging. The POS tagging is based on default pre-trained model

Given the training data in the form of a sequence of tokens, the labeling generates a label sequence as long as the token sequence. The labeling scheme is the commonly used BIO (begin, inside, outside). Three types of positive tag are "technology", "value", and "attribute", and one type of negative tag is "other". Each positive tag can be either "begin" or "inside". Therefore, in total, seven tag types were used.

4.2.2 CRFs with Tag Modification

In CRFs, the probability of a particular label sequence y given observation sequence x is assigned as a normalized product of potential functions (Wallach, H. M., 2004).

$$p(y|x,\lambda) = \frac{1}{Z(x)} exp\left(\sum_j \lambda_j F_j(y,x)\right) \quad \textbf{(4.1)}$$

In the above equation, $Z(x)$ is a normalization factor; λ_j are parameters to be estimated from training data; and

$$F_j(y,x) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, x, i) \quad \textbf{(4.2)}$$

where $f_j(y_{i-1}, y_i, x, i)$ is either a state function $s_j(y_i, x, i)$ of the label at position i and the observation sequence, or a transition function $t_j(y_{i-1}, y_i, x, i)$ of the entire observation and the labels at position i and position i - 1 in the label sequence.

Since only state functions were used, the difference of function is pertaining to the observation sequence. These observation sequences were defined as follows:

1. n-gram in the original sequence

2. n-gram in the POS tag sequence

3. current POS tag with other observed unigram and its POS tag

An n-gram is a contiguous sequence of n items from a given sequence. A unigram is an n-gram of size one. The maximum size of n-gram used is five. When unigram is adapted, the maximum distance from the observed unigram to current state is four. In other words, if the observed unigram is too far away from current state, then it was not considered in current CRFs model.

To increase positive tags, partial non-entity tags are modified into entity tag. The criterion for deciding which non-entity tag should be modified is that the CRFs model does not have an enough confidence to give the non-entity tag. The new tag is an entity tag with the highest confidence.

Formally, if the probability of the state recognized as non-entity is not high enough (that can be controlled by a threshold confidence value e.g., 90%), the non-entity tag is modified by an entity tag. An entity tag is chosen as the replacement when its probability is the maximum among that of all entity tags. In CRFs, the probability of each state given the observation sequence could be calculated as $p(y|x, \lambda)$. So the update rule is as follows:

IF

$p(Y=\text{"other"}|x, \lambda) < t$ // t is a threshold confidence value

THEN

$p(y|x, \lambda) = \max_{Y \neq \text{"other"}} p(Y|x, \lambda)$

$y := \text{argmax}_{Y \neq \text{"other"}} p(Y|x, \lambda)$

4.2.3 Pattern-based Extraction

The CRFs with tag modification does not have the capability to solve two problems. First, the length of the observation sequence is too long. In this case, some tokens, which offers indicator information but are too far away from current state, are not involved in the model. This situation is very common in patent, because the sentence in patent is usually very long due to the use of preposition phrase or parallel structure. The second problem is ambiguity. It is difficult to differentiate attribute entity from technology entity. The CRFs model only contains raw text and part-of-speech information, while both attribute entity and technology entity are usually a noun phrase. Therefore, without additional knowledge, it is difficult to make a judgment whether a noun phrase is a technology entity or an attribute entity.

To address these two challenges, some patterns are considered. First, the words expressing value entity are related in terms of meanings e.g., an adjective related to polarity opinion, namely good or bad, or a verb related to making some changes, e.g., “improve”, “facilitate”, “adjust”, “reduce” and “prevent”. Second, the words expressing attribute entity is usually a noun phrase, and the attribute entity is usually the nearest noun phrase to the value entity. For example, if the value entity equals to “improve”, “improves”, “improving” and “improvement”, then the attribute entity is usually the nearest noun phrase after the value entity; if the value entity equals to “improved”, then the attribute entity is usually the nearest noun phrase before the value entity. Thirdly, the words in value entity and the words in attribute entity should not form a collocation.

To utilize these patterns, a method was designed. This method firstly identifies candidate value entity and noun phrase. Next, for the nearest noun phrase to a value entity, it checks whether a collocation can be formed. If false, the noun phrase is identified as attribute entity and the candidate value entity is identified as value entity.

The candidate value entity was identified by a list of indicator words. Using the training data, a word list of indicator words was built. The word list was expanded by adding synonyms of every word in the list. There synonyms were from WordNet (Miller 1995; Fellbaum 1998), which is a thesaurus.

The noun phrase is chunked by a POS-based chunker. Rules are used to determine how long the noun phrase is enough to be an ATTRTBUTE.

For the collocation checking, a stopword list is built for every indicator word. To build such a stopword list manually is very difficult. So it was learned from training data and the criterion for accepting a stopword is Laplacian:

$$Laplacian = \frac{e+1}{c+e+1} \quad (4.3)$$

where c is the number of correctly matched attribute entity and e is the number of errors. If the Laplacian is smaller than 0.5, then the stopword is accepted.

4.3 Evaluation

The evaluation results in this chapter are the same as the one reported in the NTCIR-8 (Wang, Loh & Lu, 2010). The organizer of NTCIR-8 offered tagged topics for training and untagged topics for test. The evaluation was executed by the organizers.

4.3.1 Dataset

The raw text of each topic is the title and the abstract of a patent or a paper. The training data consists of 300 patent topics and 300 paper topics, while the test data is composed of 200 patent topics and 200 paper topics.

The distribution of the desired entities is shown in Table 4-1, including technology entities in title (TT), technology entities in abstract (AT), attribute entities in abstract(AA), and value entities in abstract (AV).

Table 4-1 The entity distribution

Entity Type	Patent	Paper
TT	39	93
AT	847	342
AA	213	204
AV	198	193

4.3.2 Evaluation Measures

The evaluation measures are recall, precision and F-measure (Manning, Raghavan & Schütze, 2008). F-measure (also F_1 score or F-score) is the harmonic mean of precision and recall. The precision is the number of correct results divided by the number of all returned results. The recall is the number of correct results divided by the number of results that should have been returned.

The calculation of the precision and the recall of the technology in title are given as an example. A standard list, each element of which is a technology in a topic, is built for the standard result. Similarly, a system list is built for the system result. A matched result is a result appears in both standard list and system list. The number of correct results is the number of matched results. The number of all returned results is the size of the system list. The number of results that should have been returned is the size of the standard list.

4.3.3 Results

Three system runs were submitted: NUSME-1, NUSME-2 and NUSME-3. The NUSME-1 adopted the CRFs method. Compared to the NUSME-1, the NUSME-2 added a tag modification step. The NUSME-3 enhanced the NUSME-2 by integrating the output of the pattern-based method with the output of the NUSME-2. In other word, NUSME-3 is the proposed method for E-model extraction.

The F-measure submitted by all participants in the task is shown in Figure 4-1 (patent topics) and Figure 4-2 (paper topics). The F-measure of NUSME-1, NUSME-2, and NUSME-3 are denoted with the bars filled with sparse lines, dense lines, and black color, respectively. The NUSME-2 and NUSME-3 achieved relatively good results with respect to F-measure for both patent topics and paper topics. Specially, the NUSME-3 was the best among all participants not only for patent topics but also for paper topics. Compared to NUSME-1 and NUSME-2, the NUSME-3 expended greater efforts and obtained better results. Notably, a big improvement was achieved by the tag modification step.

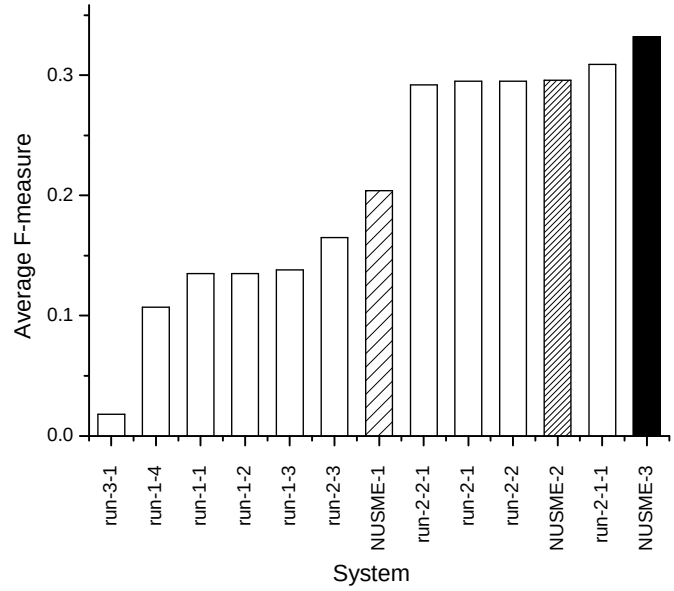


Figure 4-1 The F-measure of all systems on patent topics

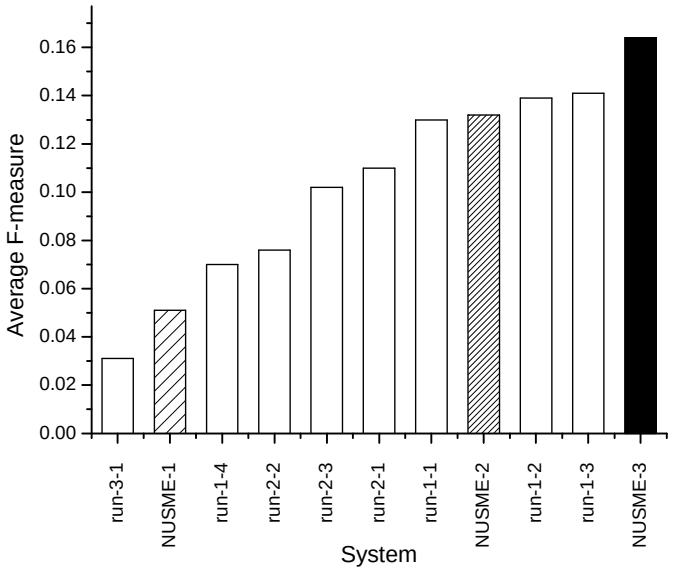


Figure 4-2 The F-measure of all systems on paper topics

The recall and precision of the three system runs using the patent data are shown in Figure 4-3 and Figure 4-4. The tag modification step, namely from NUSME-1 to NUSME-2, is able to improve the recall. It induces the CRFs model to output more positive tags, thereby increasing the chance of finding correct

entities. However, at the same time, the additional output also increases the chance of reducing the precision. That is why the precision is reduced.

In Figure 4-4, the general trend is a decrease of precision from NUSME-1 to NUSME-2 with the exception of AA. The anomaly is because no correct entity of AA was discovered in NUSME-1 and hence its precision is zero. Therefore, once a correct entity is discovered in the second run, the precision of AA could be improved.

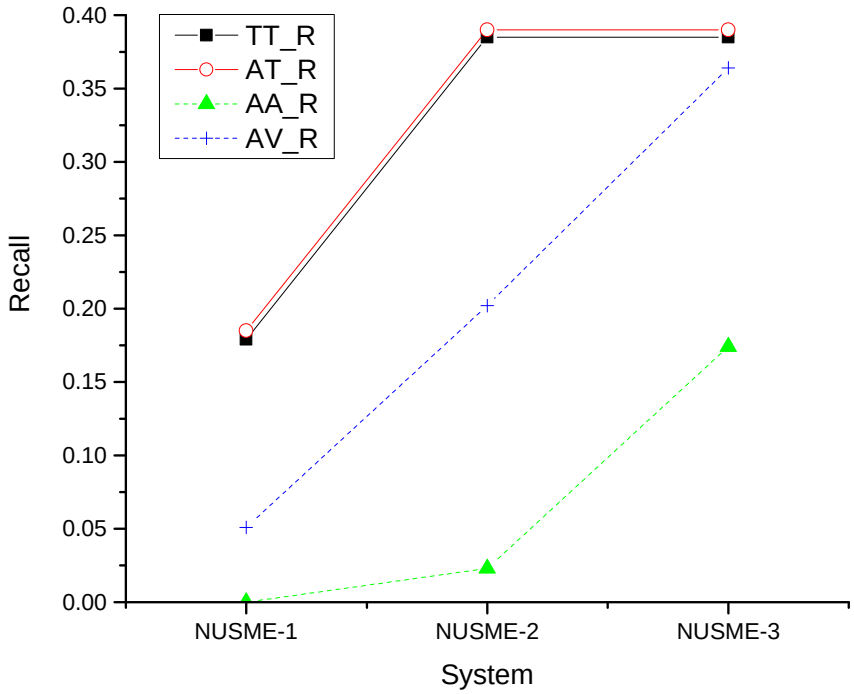


Figure 4-3 The recall of NUSME system runs on patent data

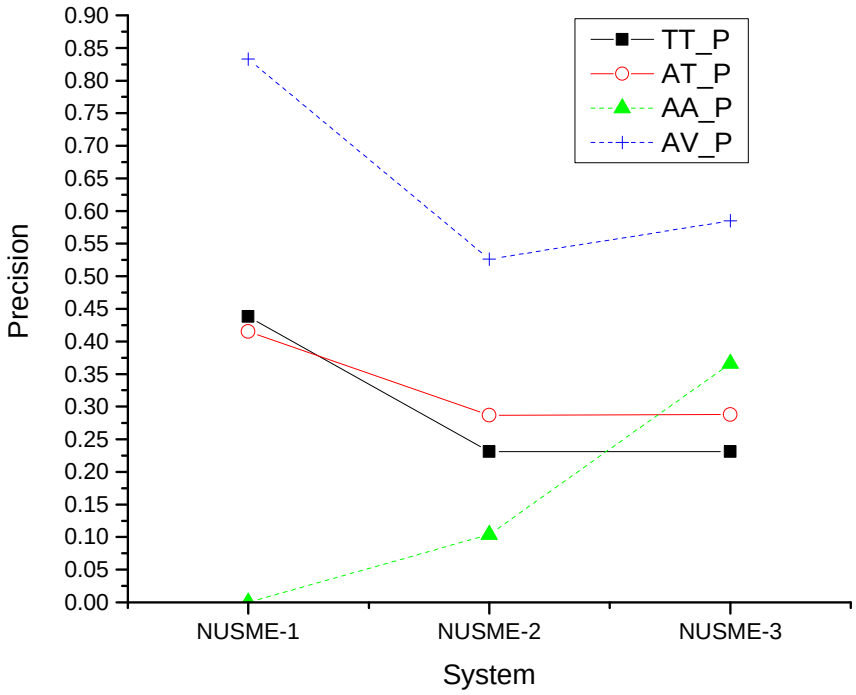


Figure 4-4 The precision of NUSME system runs on patent data

It can be observed that the manual designed patterns (from NUSME-2 to NUSME-3) had improved both recall and precision of AA and AV. That is because such patterns are designed to overcome the weakness of the built CRFs model, and usually human intelligence is superior. There is no difference on TT and AT, because the patterns adopted are all related to attribute entity and value entity, not technology entity.

In the CRFs-based method, the four entity types i.e. TT, AT, AA and AV are treated equally. However, TT and AT are quite different from AA and AV, because AA and AV, as discussed above, are relational entities i.e. they usually appear together. This important feature has not been considered in the CRFs method. The pattern-based method was designed to utilize the relations between AA and AV. Therefore, integrating CRFs with pattern-based method produces the best results.

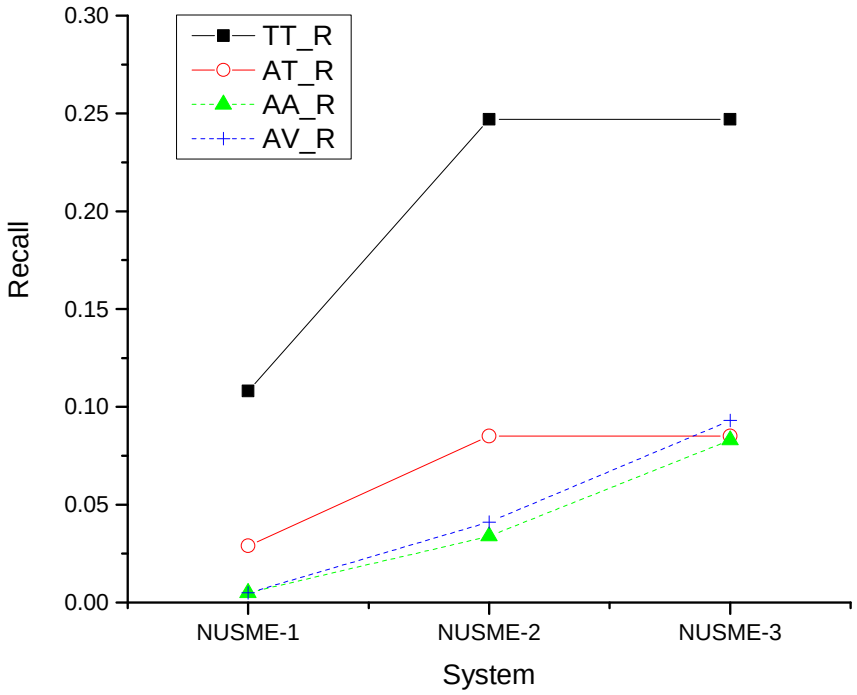


Figure 4-5 The recall of NUSME system runs on paper data

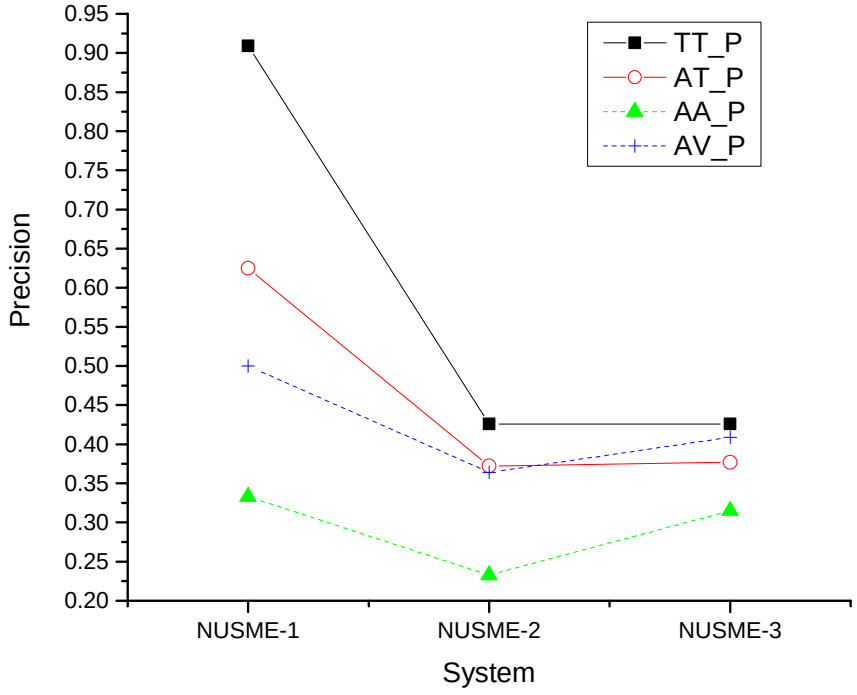


Figure 4-6 The precision of NUSME system runs on paper data

The results of paper data, which can be observed from Figure 4-5 and Figure 4-6, are almost the same as that of patent data. There is no obviously difference between patent and paper on writing the abstract and the title.

4.4 Summary

To extract E-model, a method is proposed and evaluated in NTCIR-8 patent mining task. The proposed system adopted both CRFs-based method and pattern-based method. Compared to the original CRFs method, the proposed modified CRFs module achieved a better F-measure. Moreover, the proposed pattern-based method can overcome the weakness of the CRFs-based method. A relatively good result, compared to other participants, was achieved.

Although the proposed method is relatively good, its absolute performance is not good enough. Moreover, as the first IE evaluation pertaining to technology and effect entity, the Technical Trend Map Creation task focused on entity extraction, rather than relation extraction. Although the effect entity was described by two relational entities, the final evaluation was based on individual entity. However, this task offers a benchmark for future research and a corpus for effect entity.

CHAPTER 5

EFFECT-ORIENTED SEARCH ENGINE

In previous chapter, the E-model extraction module of the effect-oriented search engine, discussed in Chapter 3, is handled as a NER problem. However, the NER method for E-model extraction is not good enough for practical use. In this chapter, an alternative method based on parsing was proposed. Moreover, this chapter introduces the entire effect-oriented search engine in detail from the lowest module to the highest module. It covers the E-model extraction with dependency parsing, query expansion, query-document matching and re-ranking. A case study is given to show the effectiveness of the proposed effect-oriented search engine.

5.1 E-model Extraction Based on Dependencies

It was observed that extracting TechnologyName in the E-model (TechnologyName, PropertyName, PropertyChange) is not a simple problem. Usually, the agent of the effect cannot be found in current sentence or current clause. Two examples are given as follows:

The cost is reduced.

... so that the cost is reduced.

In the first example, the agent is out of current sentence. In the second example, the agent is out of current clause. Moreover, even if the agent is in current sentence or current clause, coreference resolution may be required. An example is given as follows:

This improves the control reliability.

In above example, the direct agent is a pronoun “this”. A coreference resolution is needed to find what the pronoun “this” refers to.

Therefore, the TechnologyName is assumed to be known. The focus of this chapter is extracting the remaining two elements: PropertyName and PropertyChange i.e., the effect. As a dependency parsing problem, the implicit

syntactic relationship between PropertyName and PropertyChange should be known. A more elaborate investigation was carried out on the 500 patents in the NTCIR-8 Technical Trend Map Creation task in order to understand better the ways of expressing effect in natural language. The effect entities are only labeled in the abstract section. Therefore, all abstracts of the 500 patents are read manually in order to discover underlying patterns. The discovered expression manners are linked with syntactical patterns.

It was discovered that an effect is expressed through an object and its character or behavior in an object-centric view. The object corresponds to PropertyName, while a character or a behavior corresponds to PropertyChange. Generally there are two categories of effects in terms of PropertyChange: through adjective-like character or through verb-like behavior. The detailed syntactical patterns for expressing effect are given in Appendix I.

It was further discovered that the head of the PropertyChange and that of PropertyName has dependency relation, no matter what the exact syntactical relation between PropertyChange and PropertyName is. Therefore, a query-focused problem is defined for extracting E-model from dependencies.

Formally, given patent abstracts, PropertyName, and PropertyChange, the E-model extraction system should label all PropertyName and ProperyChange in the abstracts.

For example, the PropertyName is assumed to be “manpower”, the ProperyChange is assumed to be “reduce”, and the text is as follows:

Automation reduces the manpower in this factory.

The dependencies obtained by Stanford parser is as follows:

nsubj(*reduces*-2, *Automation*-1)
root(ROOT-0, *reduces*-2)
det(*manpower*-4, *the*-3)
dobj(*reduces*-2, *manpower*-4)
det(*factory*-7, *this*-6)
prep_in(*manpower*-4, *factory*-7)

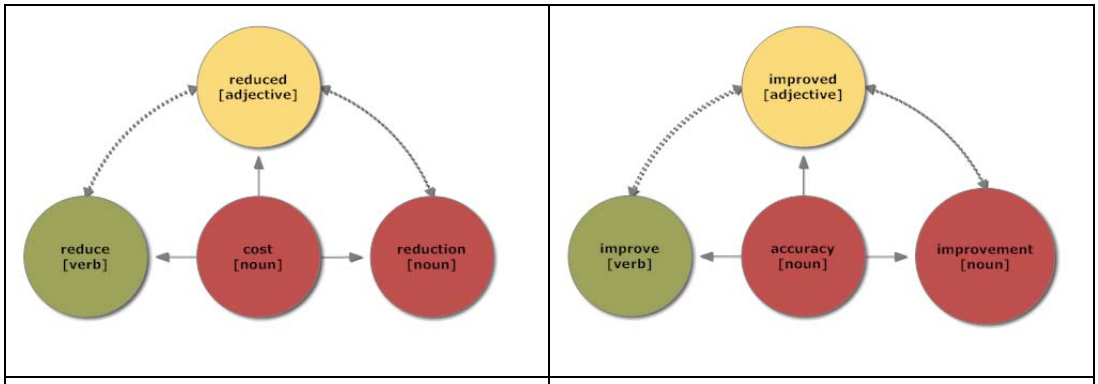
It can be observed that the fourth dependency contains both PropertyName and ProperyChange. Therefore, the “reduces” is labeled as ProperyChange and the “manpower” is labeled as PropertyName.

5.2 Query Expansion

In the above example, the given PropertyChange is “reduce”, while the labeled PropertyChange “reduces”. Although they are different in form, they are the same PropertyChange. To link the two different terms, query expansion is used.

Query expansion is the process of reformulating the seed query in order to improve the information retrieval performance. Query expansion usually includes two aspects: morphology and synonym. From the syntactical patterns in Appendix I, an additional aspect should be considered for effect-oriented search. This is because the same PropertyChange can be expressed in different ways e.g., different Part-Of-Speech (POS).

For example, expressing the decrease of the “cost” can use “reduce”, “reduction” or “reduced”, as shown in Figure 5-1(a); expressing the increase of accuracy can use “improve”, “improvement” or “improved”, as shown in Figure 5-1(b).



(a) (b)

Figure 5-1 Examples for expressing property change

For the same property, some words that express property change are semantically related. These words may belong to different POS. Therefore, this kind of query expansion is called cross-POS semantic expansion.

The implementation of cross-POS expansion and synonym expansion is based on WordNet (Miller 1995; Fellbaum 1998). In WordNet, the main relation among words is synonymy. The synonyms are grouped into sets, namely synsets. Therefore, the synonym expansion can directly utilize the synsets and the cross-POS expansion should be based on a kind of synset-to-synset relation. In WordNet, the relation between two words belong to different POS is called derivative relation. The WordNet supports derivation, but does not define derivation well. There are two pointers for derivation: "derivationally related" and "derived from adj". The former is evoked by a noun, a verb or an adjective, while the latter is evoked by an adverb. Both pointers are connections between two words, rather than two synsets. In other words, the derivation pointer is a word-to-word relation, rather than a synset-to-synset relation.

To build the synset-to-synset derivative relation, a direct search method and an indexing search method are proposed. Given a synset, the direct search method finds a set of derived synsets through the derivation relations between words. As shown in Figure 5-2, given a synset, all words contained in the synset are retrieved. The derivatives of these retrieved words are then obtained. Lastly, the synsets, which theses derivatives belong to, are retrieved. Thus, all derived synsets of the given synset is obtained.

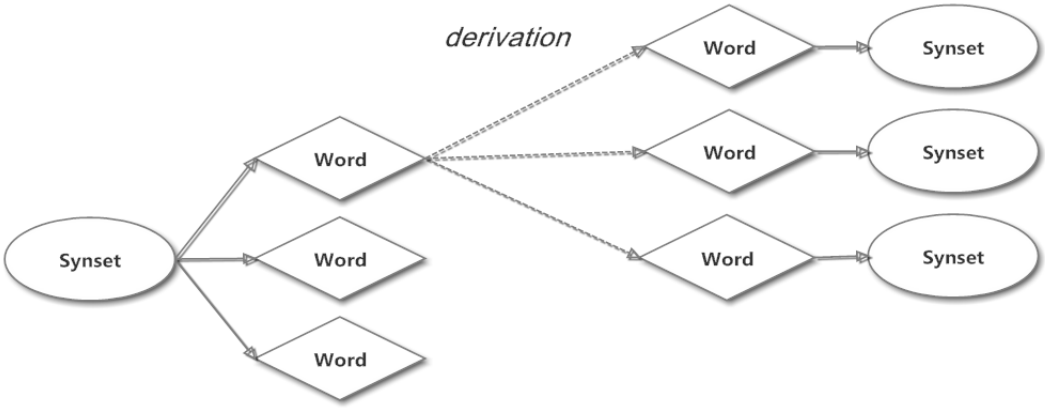


Figure 5-2 The derivation relations between synsets

The indexing search method relies on an index to retrieve the derived synsets of a given synset. The index is a sorted list, in which each element represents a pointer from a synset to a set of synsets. Thus, a binary search algorithm takes logarithmic time to locate an element. The worst case performance is $\mathcal{O}(\log n)$.

The index was built through traversing all synsets in WordNet. A direct search method was carried out for each synset. The results are checked manually, and invalid links between two synsets are removed.

The morphology expansion is implemented by an inflector, rather than a stemmer. Usually, stemming, which is the process for reducing inflected (or sometimes derived) words to their stem, base or root, is carried out in IR (Manning, Raghavan & Schütze, 2008) or NLP-based IR (Strzalkowski & Vauthey, 1992). Both query and patents should be tokenized and stemmed in the same way before a matcher can calculate the similarity between the query and the document. In contrast, inflection is the modification of a word to express different grammatical categories such as tense, person, and number. Specifically, conjugation is the inflection of verbs; declension is the inflection of nouns, adjectives and pronouns. If the stemming for the query is replaced by the inflection, both the stemming and tokenization for the patents can be removed. This leads to a saving of time.

The algorithm for inflection has two steps. In the first step, the inflection is based on a sorted exception list. A binary search is implemented. If a given word is not on the exception list, the word will be passed to the second step. In the second step, the inflection follows regular English grammar rules.

5.3 Query-Document Matching

Since parsing is time consuming, immediate response requires an offline parsing and an indexing (Strzalkowski & Vauthey, 1992). For patent database, it means an additional mass memory for storing parsing results and additional searching time. To avoid additional mass memory and searching time, online parsing is preferable. The long online parsing time can be reduced significantly by parsing partial sentences rather than all sentences. A candidate sentence for parsing should contain words relevant to the query. Therefore, a sentence filter was designed for filtering irrelevant sentences before paring.

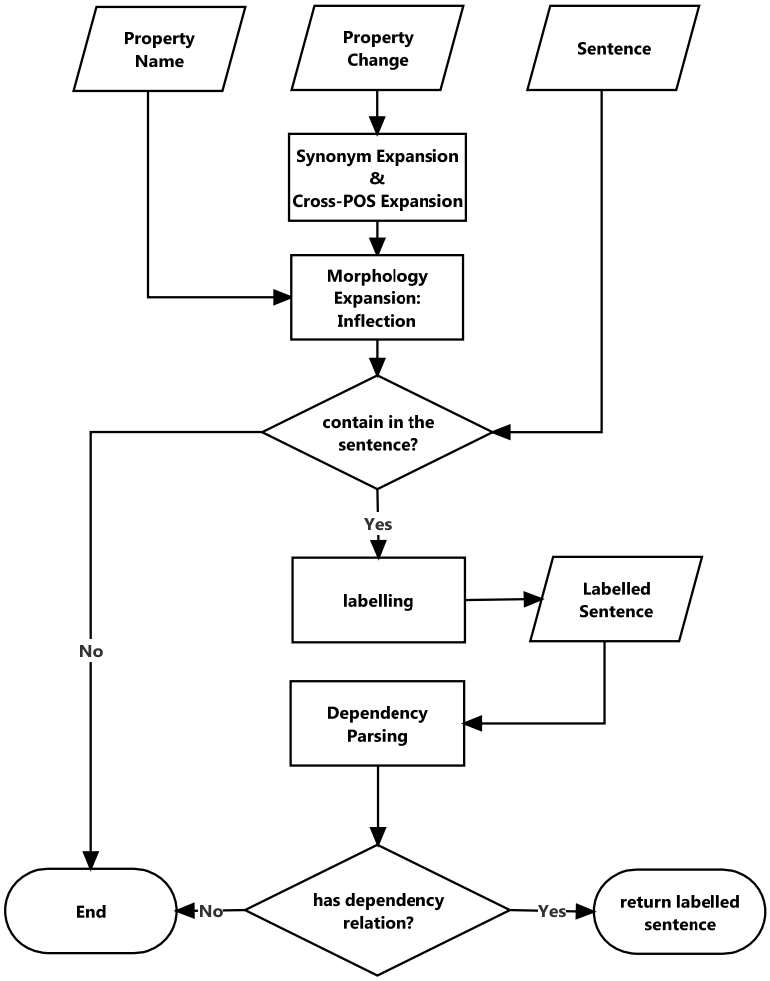


Figure 5-3 The query-document Matching

As shown in Figure 5-3, after synonym expansion, cross-POS expansion and morphology expansion, the sentence filter filters irrelevant sentence. The relevant sentences are parsed with dependency parsing.

5.4 Re-ranking

Given n ranked patents from an external search engine, the effect-inclusive relevance of patent i is calculated as follows:

$$R_e(i) = p\frac{M(i)}{\max_j M(j)} + (1-p)\frac{n+1-R(i)}{n} \qquad (5.1)$$

Here, $R(i)$ is the original rank of patent i; $M(i)$ denotes the number of matched sentence in patent i; while $p \in [0, 1]$ is the penalty factor for the effect item. The

default value of p is 0.5, which means both the original rank given by the external search engine and the matched effect information contained in a patent are equally important. In other words, a patent is relevant, if it is considered as relevant by the external engine and it contains the desirable effect information.

5.5 Search Engine System

As shown in Figure 5-4, the search process has two steps. In the first step, a structured query is input and is translated into a valid query of a third-party search engine. A conventional patent search process is evoked and a list of relevant patents is returned. The structured query is an E-model and consists of three entries: technology, PropertyName and PropertyChange. In the second step, the obtained patents are re-ranked according to the effect- inclusive relevance.

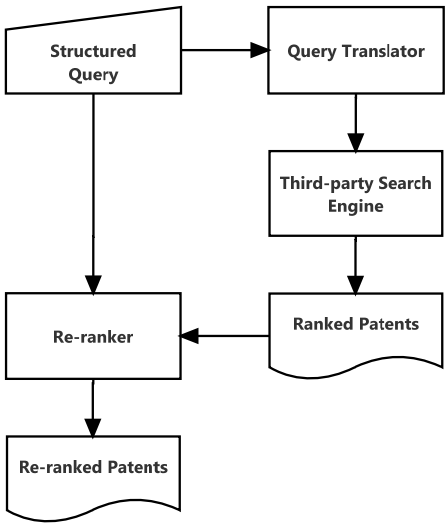


Figure 5-4 The re-ranking in the search engine

The input interface of the patent search system is shown in Figure 5-5. Like many other patent search engines, the selection of specific search field is available.

To use the effect-driven patent retrieval, a user is required to conceive his query following the logic of technology ontology. The search engine offers three input entries: product (i.e., technology), patient (i.e., property name) and relation (property change). Both “product” and “patient” are expected to be a simple noun phrase. The “relation” between the product and the patient is expected to be a single noun, verb or adjective.

Figure 5-5 The interface of the patent search engine

After clicking the "search" button, the search engine will ask the user to select the exact meaning of the relation word. This is realized by evoking the WordNet. Next, after ticking the desirable semantics and clicking the "continue" button, the search engine will return the search results. The search result is a list of patents. Most relevant patent is show first. The discovered desired relations are highlighted in the search result.

5.6 Case Study: Effect-oriented Patent Retrieval

The case in the Chapter 1 is used again. In this case, the goal is to search for patents pertaining to wireless mouse, for which the mouse does not need to change battery frequently, or has a long battery life. Naturally, the product should be "wireless mouse". The patient and the relation are assumed to be "battery life" and "long", respectively.

As shown in Figure 5-6, the search engine will suggest 12 meanings of the "long". As shown in Figure 5-7, the search result not only shows a list of relevant patents, but also highlights the discovered relations. Those patents containing the queried effect are highly ranked.

Figure 5-6 The interface of semantics selection

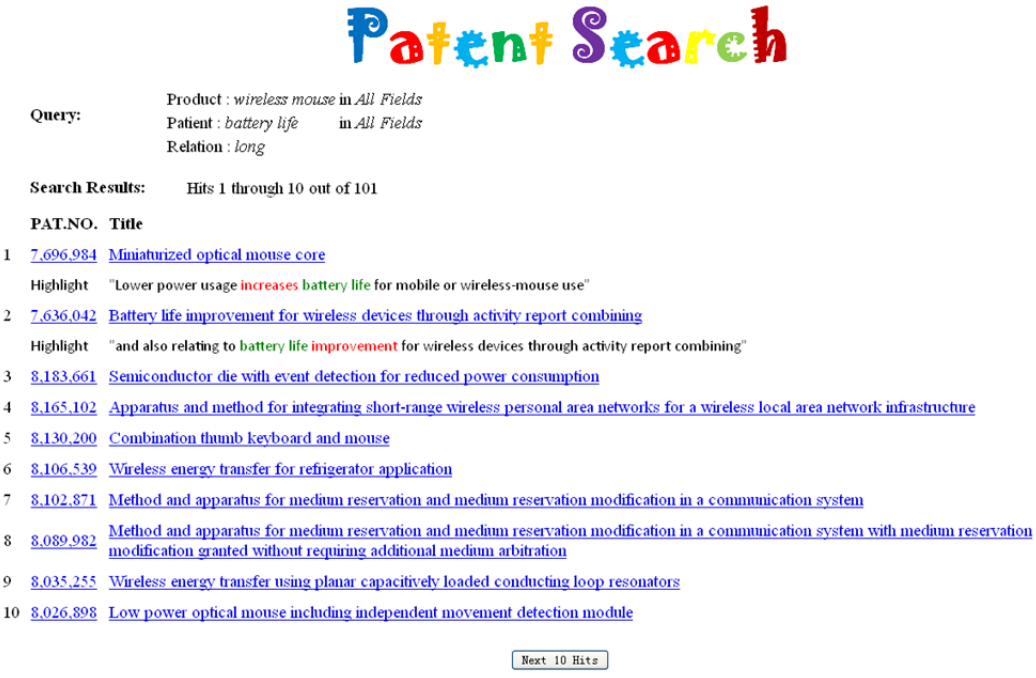


Figure 5-7 An example of search results

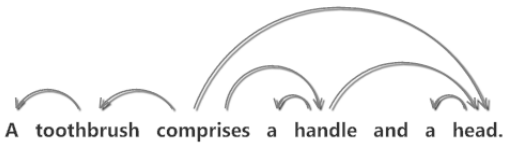
5.7 Summary

In this chapter, a method is proposed to extract E-model from dependencies. Moreover, the effect-oriented search engine, discussed in Chapter 3, is introduced in detail, including the necessity for query expansion, especially the one crossing part-of-speech, query-document matching and re-ranking. Compared to conventional search engine under term independence assumption, the effect-oriented search engine uses additional effect information as a filter to reduce the number of returned patents.

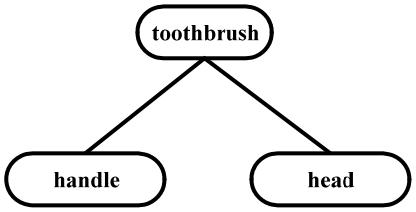
CHAPTER 6

INDEPENDENT CLAIM SEGMENT DEPENDENCY SYNTAX

The patent growth mapper, discussed in Chapter 3, has an S-model extraction module. The extracted S-models are used for supporting the patent growth mapper. To extract S-model of a patented technology from its patent's claims, the dependencies are utilized. For example, with the dependencies, as shown in Figure 6-1(a), its S-model, as shown in Figure 6-1(b), can be formed. Therefore, dependencies are required for solving the S-model extraction problem.



(a)



(b)

Figure 6-1 An example of extracting S-model with dependencies

However, as discussed previously, claim parsing is a challenge. To address this challenge, this chapter firstly gives a thorough discussion on the difference between claim syntax and dependency grammar. Moreover, practical problems of claim parsing with existing parsers were investigated.

To solve the discovered problems, new dependency syntax, called Independent Claim Segment Dependency Syntax (ICSDS), is defined for independent claims and is introduced in this chapter.

6.1 Peculiarities of Claim Syntax

The claim syntax obeys exactly the English grammar. However, it is peculiar. These peculiarities had been discussed (Parapatics & Dittenbach, 2011). In this study, the discussions focus on the inconsistency between the peculiarities and dependency grammars.

(1) Template

There are some formal templates for starting a claim. They are necessary and are used for organizing multiple claims. For examples, “We claim:” (in patent numbered US7954694) before a first independent claim; and “The file folder of claim 3, wherein” (in patent numbered US7954694) before a dependent claim, in which the “file folder” is the patented product.

Such text does not offer specific information pertaining to the patented product, but does affect dependency parsing. The counter measure is to exclude them from parsing.

(2) Complex noun phrase as sentence

A dependency-grammar-based parser may allow a noun phrase to be a sentence. For example, when the input text is a single noun, the noun is considered as a sentence. When the input text is a very simple noun phrase structure, e.g., a determiner plus a noun, the noun phrase is considered as a sentence. However, noun phrase is easy to depend on another constituent, if it exists.

In claim, it is very common that a complex noun phrase is an independent sentence, and at the same there are many other constituents. Thus, a dependency-grammar-based parser usually treats the entire complex noun phrase as a constituent of another sentence, and makes a wrong parsing. The counter measure is to allow noun phrase to directly use ROOT as the head.

(3) Tense

The basic tense in claim is present tense rather than past tense. Generally, the past tense and the past participle have the same verb form. The post attributive present participle phrase or post attributive past participle phrase is very common to form complex noun phrase. It is hard for a dependency-grammar-based parser to distinguish post attributive past participle from verb past tense, because a dependency-grammar-based parser usually prefers a sentence containing a predicate to a noun phrase. The counter measure is to execute POS correction.

(4) Parenthesis

Generally, a dependency-grammar-based parser usually treats an input text as a single sentence, and assigns dependency for every word in the text. However, a claim may not be a single sentence, because it is very common that an independent sentence is directly inserted into a claim. Thus, incorrect automatic parsing is inevitable.

(5) Recursion

Recursion is common in independent claim, especially when expressing structure information. For instance, "*wherein the body includes a graphical region comprising an ornamental three dimensional sculpture*" (in patent numbered US7917986) is best analyzed as a main sentence "*wherein the body includes a graphical region*" having an embedded sentence "*a graphical region comprises an ornamental three dimensional sculpture*". Moreover, the predicates of the main sentence and sub-sentence express the same semantics. This increases the difficulty of dependency parsing.

(6) Coordination

In dependency grammar, coordination is defined trickily. For example, in sentence "*A camera comprises a lens and a body*", the head of both "lens" and "body" should be "comprises". However, in dependency grammar, the head of "and" is assigned as "lens" and the dependency relation is assigned as "coordinator". At the same time, the head of "body" is also assigned as "lens" and the dependency relation is assigned as "conjunct". Additional step is needed to reveal the reasonable dependency relation.

Coordination is common in claim, since a product can include several components. Although the definition of coordination in dependency grammar is not a problem, too many coordination increases the difficulty of correct dependency parsing.

(7) Long Distance Dependencies

Due to above mechanisms, such as noun phrase as sentence, parenthesis, recursion and coordination, dependencies in a claim can be very long. Long distance dependencies not only increase the difficulty of correct dependency

parsing, but also require significant computational cost. The counter measure is to execute claim segmentation and build segment dependency.

6.2 Practical Problems of Direct Parsing

To have a feeling on practical problems of dependency parsing on claim with existing parser, two parsers are selected to parse a small sample dataset. One parser is the Stanford parser, while the other one is the MaltParser. A detailed parser comparison can be found at (Cer, Marneffe, & Jurafsky et al. 2010). It was said that MaltParser is much faster, while Stanford parser is much accurate. A small sample dataset of patent was collected. It contains 22 claims and 20 abstracts, in which the effect relations are manually labeled. Manual evaluation is carried out through making judgment about whether the labeled effect relations can be derived from the parsed text.

It was observed that two parsers are as good as each other when parsing abstract. The recall for both parsers is 95.00%. Mistakes were made on the same abstract, which may be too difficult to correctly parse. However, Standford parser is much better than MaltParser when parsing claim. The recall of Standford parser is 81.82%, while that of MaltParser is 77.27%. This conclusion is consistent with previous work (Cer, Marneffe, & Jurafsky et al. 2010). A more careful examination discovered that the mistakes only occur in verb-centric structure. Generally, a local relation e.g., adjective-noun relation can successfully be identified. In contrast, a non-local relation e.g., long distance dependency, usually cannot be found.

The Stanford parser was further tested due to its acceptable parsing accuracy. The test focused on computational complexity. Both space complexity and time complexity were considered.

For this study, a dataset, called PPAT273, is built manually. In PPAT273, there are a total of 273 product patents, which were downloaded from United States Patent and Trademark Office (USPTO). Each patent is a utility patent and describes a whole product. There are ten product types, including toothbrush, digital camera, razor, lighter, forceps, file folder, mobile phone, surgical scalpel, hypodermic needle and paper punch.

From PPAT273 dataset, 273 first independent claims (referring as claim in the rest of this chapter) were extracted. The length represents the number of tokens in a text string. The length of a claim is defined as the number of tokens it contains. The statistical result is shown in Figure 6-2. It is observed that the length of most claims is more than 100. At the extreme, the length of a claim may exceed 800.

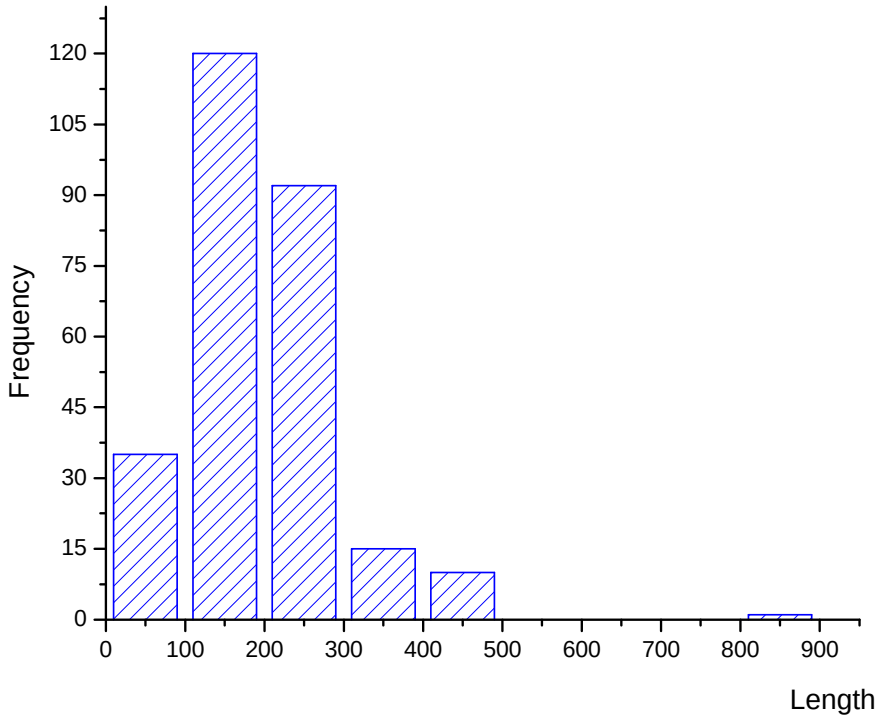


Figure 6-2 The frequency of length

It is reported (on the Stanford parser's homepage) that the memory use is proportionally the square of the length. Generally, parsing a text with length 20, 50, and 100 needs approximately 250MB, 600MB and 2100MB, respectively. Therefore, the Stanford parser is unable to parse most claims in the PPAT273 dataset on a common personal computer, of which the maximum memory is 2000MB. This conclusion is consistent with previous work (Parapatics & Dittenbach, 2011), which only tried physical memory heap size no more than 1000MB. In this study, it was tested and found that 700MB memory can only parse a text with length no more than 28. That is worse than the expected. However, when the memory is increased to 1400MB, the parser can parse a text with length up to 206. This means more than half of the claims in the PPAT273

dataset can be parsed. It seems that when the memory is added to a high enough value, parsing does not require the memory size as much as the expected one, which is proportional to the square of the length. It is also expected that high performance computing server or cloud computing can offer the capability to parse a very long claim whose length is more than 800.

Compared to space complexity, time complexity is more important. To test the parsing time, six sample claims were selected from the 273 claims. The lengths of five claims are evenly distributed in a range from 0 to 250, with 50 as the interval. The sixth claim is the shortest one whose length is 21 among the 273 claims. For each claim with length *l*, it was parsed *l* - 10 times. In the first time, the entire token sequence of the claim is passed to the parser. In the next time, the last token in the token sequence is removed. The cutting is repeated until the length of the token sequence equals to 10.

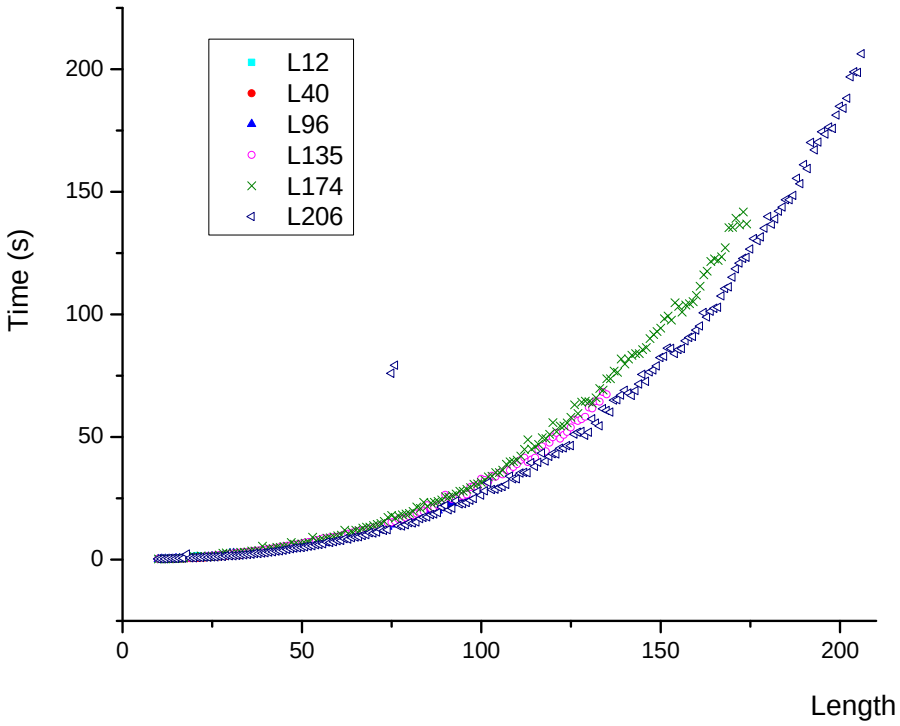


Figure 6-3 The relation between length and time

The test results are shown in Figure 6-3, it was observed that generally the parsing time is monotonically increased with the increase of length. When the length is less than 50, the increase of parsing time is not significant. Parsing a 50 long claim requires about five second. However, the parsing time increases sharply when the length is more than 100. Parsing a 140 long claim needs more

than one minute; parsing a 170 long claim needs two minutes; while parsing a 200 long claim needs three minutes.

6.3 Basic Idea of ICSDS

To hand long length and Long Distance Dependencies, one way is to execute claim segmentation. To maximize the utilization of existent natural language resources, every segment is parsed with an existent the parser. In other words, it is assumed that a claim can be segmented in a way that most word-to-word dependencies in each segment can be correctly parsed with a conventional parser. A higher-level parser further parses segment-to-segment dependencies and builds the word-to-word dependencies that are crossing segments.

Generally, the Independent Claim Segment Dependency Syntax (ICSDS) is dependency-based syntax designed for parsing independent claims, which cannot be directly parsed well with traditional dependency grammars, e.g., the standard Stanford dependencies. It belongs to a class of modern syntactic theories that are all based on dependency relation. It includes means for segmenting an independent claim into segments, recognizing segment features, building segment dependencies and assembling segment dependencies with word-to-word dependencies.

6.4 Properties of ICSDS

Apart from all the words in a claim, an additional token is defined as ROOT, which means the root of the parsing tree. The properties of the ICSDS include:

(1) Connectivity

All the words are connected with the dependency relations.

(2) Single Head

Apart from ROOT, each word must have and can only have one head.

(3) Partial Planarity

Apart from the dependency relation connecting ROOT, a dependency relation does not cross any other dependency relations when drawn above the words.

(4) Proximity Principle

Each dependent depends on the closest possible head.

6.5 ICSDS parser

Without large training dataset, this study focuses on grammar-based parsing method. The first implementation of the ICSDS is based on the Stanford parser. The system overview is shown in Figure 6-4. Since loading a trained Stanford parser requires many seconds, the ICSDS parser processes claims in a manner of batch processing.

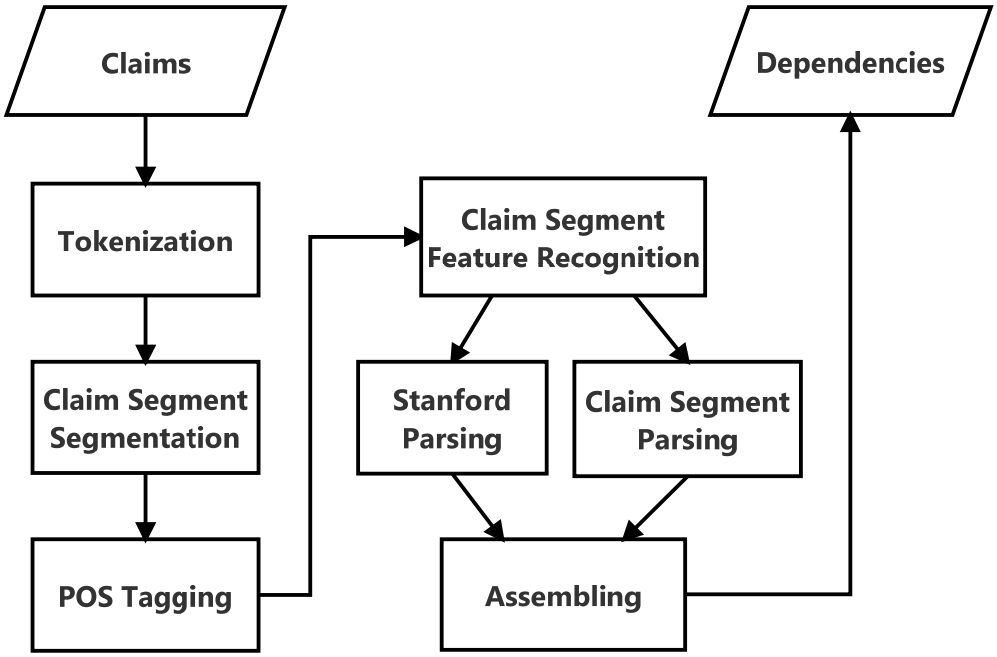


Figure 6-4 The system overview of the ICSDS parser

6.5.1 Tokenization and POS Tagging

The tokenization and POS tagging is similar to the one in (Wang, Loh & Lu, 2010). The tokenization is completed by the Stanford tokenizer, while the POS tagging is completed by the Stanford POS tagger. Thus, the mistakes caused by using different tokenization method or POS tagging method should be minimized.

6.5.2 Claim Segment Segmentation

Given a string of tokens, the claim segment segmentation returns a sequence of claim segments. A delimiter is a mark which fixes the boundary of a segment.

The delimiter is formed by some separators. Since ICSDS prefers natural separation of text, any mark that helps separating an independent claim and making the meaning clear is considered as a separator. These known separators belong to three categories: HTML element, sequential number, and punctuation mark. Generally, two separators belonging to the same category do not occur consecutively. In contrast, two or three separators belonging to different categories may occur consecutively. Therefore, a delimiter is defined as a triple in the form of (HTML-element, sequential-number, punctuation-mark). For example, a part of the first independent claim of patent numbered US4027510 is shown as follows:

> *1. A forceps instrument comprising in combination,*
>
> *

a. an outer sleeve member,*
>
> *

b. a guiding viewing-tube support, tubular in shape, and mounted concentrically within said outer sleeve,*
>
> *

c. a tubular barrel mounted within said outer sleeve substantially concentrically around and axially slidable along said guiding viewing-tube support,*
>
> *...*

Here, the first segment is “A forceps instrument comprising in combination”, followed by the first delimiter (“br”, “TypeE”, “,”). The first delimiter contains a HTML element i.e.,
 (formally
), a sequential number of type E (see Appendix II for details), and a punctuation mark i.e., a comma. The third segment is “a guiding viewing-tube support”, followed by the third delimiter (-, -, “,”). The third delimiter contains only a punctuation mark i.e., a comma.

6.5.3 Claim Segment Feature Recognition

Given a claim segment, the claim segment feature recognition recognizes features at the starting portion and the ending portion of the input claim segment. A segment is characterized by its starting portion and ending portion. Therefore, segment feature recognition focuses on the starting portion and the ending portion of a segment.

A rule-based method is executed. Rules are created manually to support the recognition. The structure of a rule for starting portion is the same as that for ending portion. The basic elements composing a rule include segment length,

lexicon, part-of-speech (POS) and some word classes that are specially defined. For example, starting portion rule "NP,2,IA,!POS:adjective" means if a segment with length two, starting from an IA i.e., indefinite article, and the second token is not an adjective, then the segment should start from a NP i.e., noun phrase.

6.5.4 Claim Segment Parsing

Given claim segments with features, the claim segment parsing returns claim segment dependencies. If a claim segment relies on another segment to form a sentence, then there exists a dependency relation between them, while the former is called as dependent and the latter is called as head. If a claim segment does not rely on any segment to form a sentence, then its head is the ROOT. This dependency relation between two segments is a little different from that of two words.

Current implementation of the claim segment parsing adopts a rule-based method. Two major elements of the rule-based method are dependency rule and dependency constraint. The dependency rules and the dependency constraints are working together to support correct parsing. A dependency rule describes the features of both the dependent and its possible head. The adopted features include relative position, relative distance, starting feature, ending feature, and punctuation feature. Moreover, a dependency rule can include heritage. In other word, a dependency rule may allow a dependent to inherit another dependent's head. The default head is the "ROOT". Therefore, if no rule applicable, "ROOT" will be assigned as the head. Dependency constraints are used to provide additional requirements on rule matching. A dependency relation is accepted, only if a rule is matched and is subject to all constraints.

For example, four dependency rules are given as follows:

AND ←SNP
AND ←SP
NP ←SNP
NP ←SP

Here, the "SNP", "SP", "NP" and "AND" are segment features. The "NP" means noun phrase. The "SNP" means first noun phrase of the sentence. The "SP" means an inside incomplete sentence. The "AND" means "and".

As shown in Figure 6-5, a claim consists of two independent sentences. A sentence with “SP” is inserted into a sentence with “SNP”. The dependency relation of two segments is depicted via an arc with an arrowhead towards the head. It is assumed that the parser has successfully parsed all segments before the segment with “NP” above the black triangle. Thus, according to the rule “NP ← SP” and the proximity principle, this segment should depend on the segment with “SP”. Next, according to the rule “AND ← NP” and the proximity principle, the next segment with “AND” should also depend on the segment with “SP”.

A dependency constraint on coordinating conjunction can reject the first dependency relation. Briefly, a head cannot accept dependent, if its last two dependents are starting with “AND” and “NP”, respectively. Thus, current segment with “NP” will depends on the segment with “SNP” correctly, according to the rule “NP ← SNP”.

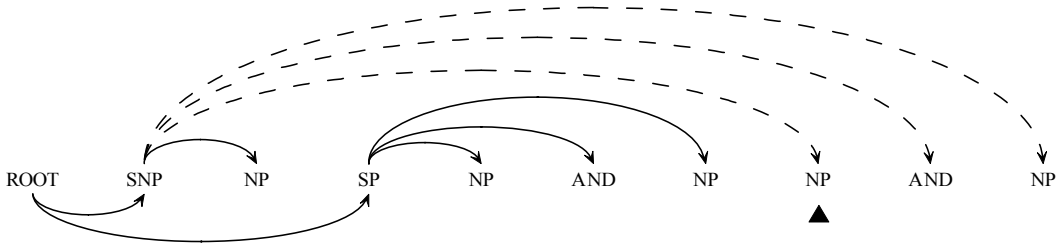


Figure 6-5 An example for explaining dependency rules and constraints

Consequently, a dependency constraint on partial planarity can reject the second dependency relation. The search for the head of the segment with “AND” will omit any segments before it, apart from the segment with “SNP”.

A left-to-right parsing algorithm is designed to read the entire segmented claim, and then identify the head of each segment from the left side of the claim to the right side. The pseudo-code is shown as below:

Algorithm: PARSE

```
01 indexOfHead ← Ø
02 foreach current segment s_c in S do
03 |  getHead ← false;
04 |  indexOfHead[s_c] ← 0;
05 |  rule ← PICKRULE(Rules, GETTYPE(s_c));
06 |  foreach segment s_i that i < c (or i > c) in terms of rule do
07 |  |  if EXAMINE(s_i) then
08 |  |  |  if MATCH(rule, GETTYPE(s_i), GETTYPE(s_c)) then
09 |  |  |  |  getHead ← true;
10 |  |  |  |  head ← GETHEAD(r);
11 |  |  |  |  indexOfHead[c] ← index;
12 |  |  |  |  break;
13 return indexOfHead
```

When a segment is in the process of head identification, it is called current segment. The head of current segment is assigned as "ROOT" initially (in line 04). In the following head search process, a rule corresponding to current segment is picked (in line 05). According to this picked rule, either the leftward segments or the rightward segments are examined one by one. For each segment under examination, the algorithm first examines dependency constraints (in line 07). If the examined segment is feasible and it together with current segment can match the picked rule (in line 08), the head in the rule (in line 10) and its actual index (in line 11) will be assigned to current segment.

6.5.5 Assembling

Given segment-to-segment dependencies, word-to-word dependencies within each segment, the assembling builds word-to-word dependencies crossing segments and returns all word-to-word dependencies. Only two kinds of word-to-word dependencies crossing segments will be assigned: verb-noun relation and adjective-noun relation, since they are necessary for S-model extraction. Given two segments, it builds a dependency relation between two words, each of which belongs to one of the two segments.

Briefly, the assembling step merges two kinds of word-to-word dependencies together. A rule-based method was used.

6.6 Examples of ICSDS Parsing

To give an intuitive feeling of the parsing result, an example is given below. The original claim is:

> *A mobile phone, comprising: a body having a ground portion; a metallic cover detachably coupled to the body, the metallic cover forming an exterior surface of the mobile phone; and a grounding unit configured to electrically connect the ground portion of the body to the metallic cover when the metallic cover is coupled to the body, the grounding unit being disposed on one of facing surfaces of the body and the metallic cover, wherein the grounding unit includes: an attachment portion located on an inner surface of the metallic cover facing the body; and an elastic extension portion extending from the attachment portion towards the body.*

In the original claim, there are 10 segments and three sentences. In the first sentence, a mobile phone (in Segment 1) comprises (in Segment 2) a body (in Segment 3), a metallic cover (in Segment 4) and a grounding unit (in Segment 6). The second sentence further elaborates the metallic cover (in Segment 5). The third sentence further elaborates the grounding unit (in Segment 7) and it includes (in Segment 8) an attachment portion (in Segment 9) and an elastic extension portion (in Segment 10). The parsing result, where the word-to-word dependencies obtained by the Stanford parser are omitted, is shown in Figure 6-6:

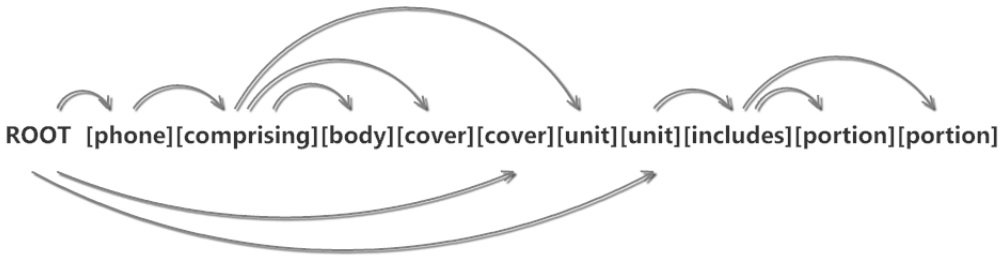


Figure 6-6 An example of the ICSDS parsing

6.7 Evaluation

Both effectiveness and efficiency of the ICSDS parser was tested. The effectiveness was test on an S-model extraction problem. The PPAT273 dataset, in which standard S-models are manually built, was used for the test. The training set consists of 173 patents, while the test set consists of 100 patents. The accurate rate is used as the evaluation measures. A parsing tree is considered as accurate, if

the S-model formed from the parsing tree is the same as the standard S-model. Both Stanford parser and the ICSDS parser were tested.

The evaluation result showed that the accurate rate of the Stanford parser was 14%, while the accurate rate of ICSDS parser was 68%. Although 68% is not very high, it is much higher than 14%.

The efficiency was evaluated through memory use and parsing time. The ICSDS parser requires less memory than the Stanford parser, because its segmentation strategy reduces the maximum length of input text. All claims can be parsed under a computer with 1.60 GHz CPU and up to 1.4 GB Java memory.

To test the parsing time, 174 claims in the PPAT273 that can be parsed with both the ICSDS parser and the Stanford parser were used. The range of length is from 26 to 210. The comparison of parsing time is shown in Figure 6-7. Apart from the shortest claim, the ICSDS parser is faster than the Stanford parser. Moreover, the variation of parsing time with the ICSDS parser is small. The range of parsing time is from 1 to 31 seconds. The parsing time with ICSDS parser is almost independent from the length of claim, when the claim length is no more than 210.

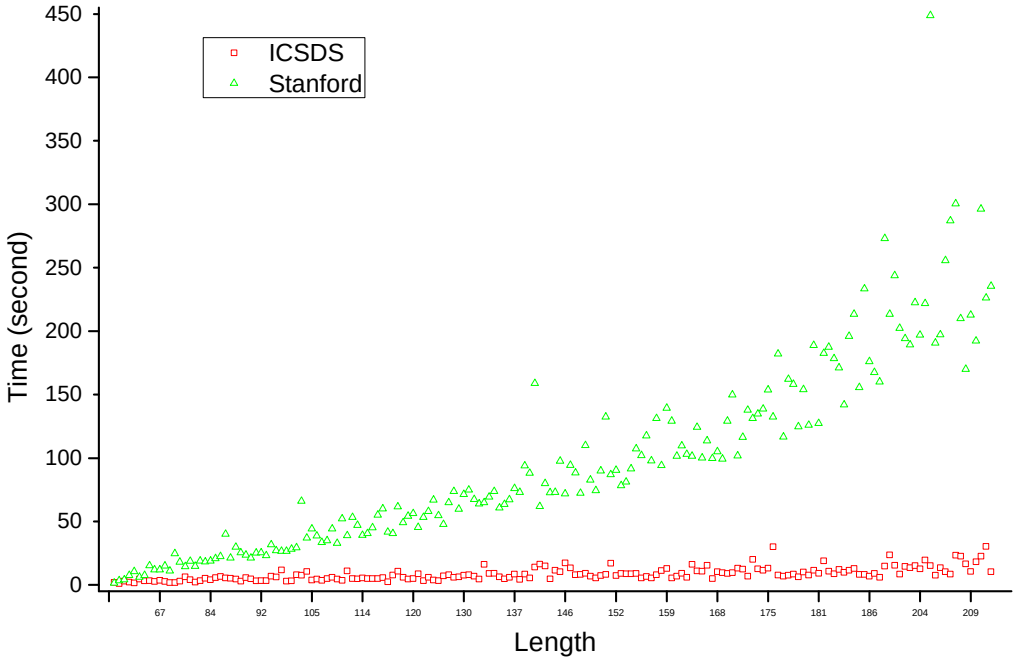


Figure 6-7 The comparison of the parsing time

6.8 Summary

This chapter discussed the peculiarities of clam syntax and the problem they caused on dependency parsing. Moreover, two famous dependency parsers were tested on claim dependency parsing. The test results show that both accuracy and speed are challenges to successful claim parsing. Fortunately, available dependency parsers demonstrated efficiency and effectiveness, when the length of the claim is short and the dependency relation is local.

Therefore, a strategy combining segmentation and assembly may be helpful. In this strategy, available dependency parsers are expected to accurately and speedily parse all segments, while these segments are accurately linked by an extra higher-level parser. Such a parser is expected to be not only more effective, but also more efficient. For example, the parsing time for a claim with length of 140 is about 60 seconds. If the claim can be segmented into three segments, each of which is less than 50, then the total parsing times is about five seconds times three i.e., 15 seconds. If the higher-level parser can correctly assemble the three segments in 45 seconds, then the whole parser is more efficient than the initial parser.

The strategy is implemented by the ICSDS parser, in which the Stanford parser is embedded. This design maximizes the utilization of available natural language technologies and resources, and reduces the effort for implementation of the new syntax. The evaluation results show that, compared to the Stanford parser, the ICSDS parser is much effective and efficient on S-model extraction.

CHAPTER 7

GRAPH SIMILARITY MEASURES

In previous chapter, the S-model extraction module of the patent growth mapper, discussed in Chapter 3, is discussed. The extracted S-models are utilized in the similarity measures module of the patent growth mapper. The similarity measures module is used for comparing technologies and measuring the difference of technologies. In this chapter, graph similarity measures for S-model were proposed. They focus on node similarity rather than treating node similarity and edge similarity equally, and keep initial intuitive similarity judgment made by human. The effectiveness of the proposed graph similarity measures was demonstrated by a few graph examples. Moreover, the recommended graph similarity measure i.e., iterative node-to-node scoring was evaluated by a real world classification problem.

7.1 Graph Representation

Formally, a graph $G = G(V, E)$, is represented as a node-node adjacency matrix. A vertex denotes a technology or a component of the technology that is described by a set of terms i.e., $v_i = \{term1, term2, \ldots\}$ and an edge denotes a inclusion relation between two vertices. If the cardinality of V is n, then the adjacency matrix $\mathcal{A}$ of this graph is an $n \times n$ matrix, in which entry $[\mathcal{A}]_{ij}$ is equal to 1 if and only if $(i, j) \in E$, and is equal to 0 otherwise. Since the direction of the edge is not considered, the adjacency matrix of such a graph is always symmetric.

7.2 Graph Similarity Scoring

The graph similarity scoring has two steps. In the first step, a node-to-node similarity is obtained. In the second step, a graph matching is executed. Formally, in the node-to-node similarity matrix X, the x_{ij} denotes the node similarity score between node i in graph G_B and node j in graph G_A. With the node-to-node similarity matrix X, the task of graph matching is to search an optimal matching between the two graphs. What often sought in the graph matching problem is an assignment matrix P. If set B has m elements and set A has $n \geq m$ elements, then P

will be an $m \times n$ matrix of only 0's and 1's, with a single 1's entry on each row, and no more than a single 1's entry in each column. If $P_{ij} = 1$, then element i of B is matched to element j of A. The graph matching problem can be solved by the Hungarian algorithm. The Hungarian algorithm calculates a maximum weight matching between two sets, each with n elements in $\mathcal{O}(n^3)$ time (Kuhn, 1955). The cost matrix $C = E - X$. The matrix E is a matrix with one in all the elements.

The final graph-to-graph similarity is the sum of node-to-node similarity scores of matched graphs, averaging by the number of nodes in the smaller graph, times the ratio of the number of nodes in the smaller graph to that in the bigger graph.

$$SIM(A, B) =$$

$$\left(\sum_{P_{ij}=1} X_{ij} / |v_B| \right) \cdot \left(|v_B| / |v_A| \right) = \sum_{P_{ij}=1} X_{ij} / |v_A| \qquad (7.1)$$

In the computation of the node-to-node similarity of two graphs, both initial node-to-node semantic similarity and topology of every graph are considered. Given node v_A and node v_B, each of which is described by a set of terms, then semantic similarity between two nodes is defined as a degree of term matching:

$$SIM_{Semantic}(v_A, v_B) \equiv (v_A \cap v_B) / (v_A \cup v_B) \qquad (7.2)$$

The topologies of the two graphs are combined with the semantic similarity in two ways. They are named as weighted node-to-node scoring and iterative node-to-node scoring, respectively.

7.2.1 Weighted Node-to-Node Scoring

The weighted node-to-node scoring calculates the similarity between two nodes as the sum of the semantic similarity and the topological similarity. The

topological similarity is defined as a function of absolute difference between the number of adjacency edges of one node and that of the other one, shown as below:

$$SIM_{Topological}(v_A, v_B) \equiv 1 / (1 + | e(v_A) - e(v_B) |) \quad (7.3)$$

Here, $e(v)$ denotes the number of adjacency edges of the node v. Thus, the weighted node-to-node similarity is calculated as follows:

$$SIM_w \equiv (1\text{-}w) \cdot SIM_{Topological} + w \cdot SIM_{Semantic} \quad (7.4)$$

Here, w is the weight of semantics similarity and is in the range from zero to one. Finally, $x_{ij} = SIM_w(i, j)$.

7.2.2 Iterative Node-to-Node Scoring

The iterative node-to-node scoring uses an iterative update method to calculate similarity. The basic idea is: a node in graph G_B is like a node in graph G_A, if they and their adjacent nodes are similar. This basic idea can be expressed as follows:

$$x_{ij}(k) = \sum_{B(ip)=1} \max_{A(jq)=1} x_{pq}(k-1) + x_{ij}(k-1) \quad (7.5)$$

The total number of summed terms is controlled by using maximum, because the sum of all terms is too big. Since each x_k is updated by a sum of several x_{k-1}, the normalization factor for each sum might be the number of summed x_{k-1}. The normalization factor matrix N can be represented as follows:

$$N = \begin{bmatrix} | & | & \dots & | \\ sum(B) & sum(B) & \dots & sum(B) \\ | & | & \dots & | \end{bmatrix} + E \quad (7.6)$$

Here, sum(·) treats the columns of the matrix as vectors, returning a row vector of the sums of each column. Matrix E is a matrix with one in all the elements.

To keep the initial relative semantic similarity, the $x_{ij}(0)$ is added to the final x_{ij}, and the sum is normalized by two. Therefore, the final score is as follows:

$$x_{ij} = (x_{ij} + x_{ij}(0)) \,./\, 2 \tag{7.7}$$

The update stop criteria can be set as a required number of runs, e.g., 1000 times, or an upper limit for the difference between x_k and x_{k-1}, e.g., 0.0001.

7.3 Examples of Graph Similarity Measures

To demonstrate the general effectiveness of the proposed similarity measures, a simple graph similarity problem is created as an example. As shown in Figure 7-1, graph (a) is compared with other eight graphs.

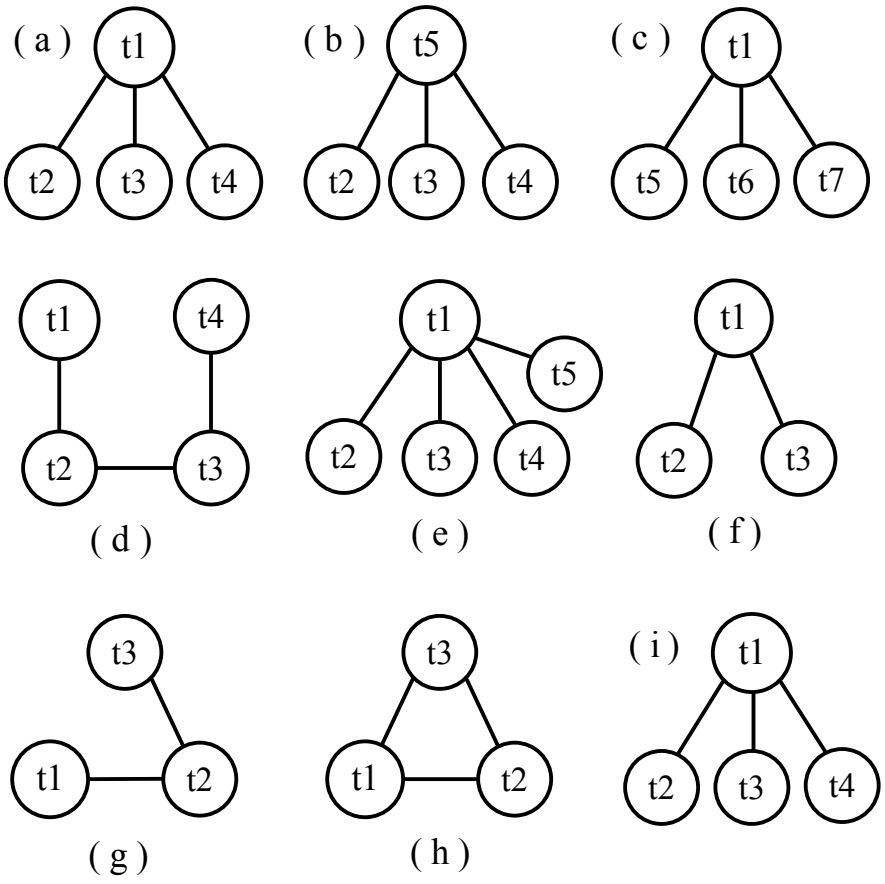


Figure 7-1 Nine example graphs. A circle denotes a node. A line denotes an edge. A "t#" in a circle denotes a term labeled on the node.

Intuitively, the expected similarity scores should satisfy requirements as follows:

(1) $1 > SIM(a,b) > 0.5$. Although root of graph (a) and that of graph (b) is different literally, they have identical components.

(2) $0 < SIM(a,c) < 0.5$. Although root of graph (a) and that of graph (c) is identical literally, they do not have any identical components.

(3) $1 > SIM(a,d) > 0.5$. Although the topology of graph (a) and that of graph (d) are different, they have the same components

(4) $1 > SIM(a,e) > 0.5$. Graph (a) is the major part of graph (e).

(5) $1 > SIM(a,f) > 0.5$. Graph (f) is the major part of graph (a).

(6) $SIM(a,f) > SIM(a,g) > SIM(a,h)$. That is because they have the same nodes, but the topology of graph (f), graph (g) and graph (h) is a tree, a line and a ring, respectively.

(7) $SIM(a,i) = 1$. That is because graph (a) and graph (i) are identical.

In VSM, the nine graphs can simply be represented as nine vectors, as shown in Table 7-1.

Table 7-1 Nine graphs in VSM

term / graph	t1	t2	t3	t4	t5	t6	t7
(a)	1	1	1	1	0	0	0
(b)	0	1	1	1	1	0	0
(c)	1	0	0	0	1	1	1
(d)	1	1	1	1	0	0	0
(e)	1	1	1	1	1	0	0
(f)	1	1	1	0	0	0	0
(g)	1	1	1	0	0	0	0
(h)	1	1	1	0	0	0	0
(i)	1	1	1	1	0	0	0

The similarity of graph (a) and other eight graphs in terms of two commonly used similarity scorings: cosine similarity and Euclidean distance (Manning, Raghavan & Schütze, 2008) are shown in Table 7-2. Scores that satisfy requirements are shown in bold. The ranking of graph (b) to (i) in terms of similarity to graph (a) with cosine similarity is the same as that with Euclidean distance. Without considering the topology, graph (a), graph (d) and graph (i) are equal. Graph (f), graph (g) and graph (h) are also identical. Overall, cosine

similarity only satisfies half of requirements. The ambiguousness caused by the Euclidean distance is more severe.

Table 7-2 The similarity comparison with VSM

Compared Graphs	Cosine Similarity	Euclidean Distance
(a) (b)	**0.750**	1.414
(a) (c)	**0.250**	2.449
(a) (d)	1.000	0.000
(a) (e)	**0.894**	1.000
(a) (f)	**0.866**	1.000
(a) (g)	0.866	1.000
(a) (h)	0.866	1.000
(a) (i)	**1.000**	0.000

Given different weight *w*, the results of the weighted node-to-node scoring method is shown in Table 7-3. Scores that satisfy requirements are shown in bold.

Table 7-3 The similarity scores based on weighted node-to-node scoring

	Graph Pair							
w	**(a, b)**	**(a, c)**	**(a, d)**	**(a, e)**	**(a, f)**	**(a, g)**	**(a, h)**	**(a, i)**
0	**1.000**	1.000	**0.750**	**0.700**	**0.625**	0.625	0.375	**1.000**
0.1	**0.975**	0.925	**0.725**	**0.710**	**0.638**	**0.612**	**0.413**	**1.000**
0.2	**0.950**	0.850	**0.700**	**0.720**	**0.650**	**0.600**	**0.450**	**1.000**
0.3	**0.925**	0.775	**0.708**	**0.730**	**0.662**	**0.587**	**0.487**	**1.000**
0.4	**0.900**	0.700	**0.750**	**0.740**	**0.675**	**0.575**	**0.525**	**1.000**
0.5	**0.875**	0.625	**0.792**	**0.750**	**0.688**	**0.604**	**0.562**	**1.000**
0.6	**0.850**	0.550	**0.833**	**0.760**	**0.700**	**0.633**	**0.600**	**1.000**
0.7	**0.825**	**0.475**	**0.875**	**0.770**	**0.712**	**0.662**	**0.638**	**1.000**
0.8	**0.800**	**0.400**	**0.917**	**0.780**	**0.725**	**0.692**	**0.675**	**1.000**
0.9	**0.775**	**0.325**	**0.958**	**0.790**	**0.738**	**0.721**	**0.712**	**1.000**
1	**0.750**	**0.250**	1.000	**0.800**	**0.750**	0.750	0.750	**1.000**

It could be observed that the weighted node-to-node scoring method can satisfy all requirements, when *w* was set as 0.7, 0.8 and 0.9. It means that involving topology can obtain more accurate similarity, but semantics plays a more important role than topological. In other words, similarity measure should consider the semantics as the primary part and the topology as the supplemental part. That can explain why VSM, which does not consider topological information, could offer acceptable results in Information Retrieval.

The proposed iterative node-to-node scoring meets all requirements discussed above, as shown in Table 7-4. Moreover, the iterative node-to-node scoring does not predefine any parameter, unlike the *w* in the weighted node-to-node scoring

method. Therefore, iterative node-to-node scoring was recommended in future graph similarity calculation.

Table 7-4 The similarity scores based on iterative node-to-node scoring

Compared Graphs	Similarity	Epoch
(a) (b)	0.675	16
(a) (c)	0.325	9
(a) (d)	0.8	30
(a) (e)	0.8	17
(a) (f)	0.75	16
(a) (g)	0.589	16
(a) (h)	0.563	26
(a) (i)	1	16

7.4 Evaluation of Iterative Node-to-Node Scoring

The effectiveness of the proposed graph model and the proposed iterative node-to-node scoring was further evaluated through a real world classification problem.

7.4.1 Experimental Setup

The PPAT273 dataset was used. The S-model of the technology in every patent was annotated manually. The classification problem is designed as a binary classification. In each class, the products belonging to this class are labeled as positive; otherwise negative. To separate training set and test set, patents are sorted in terms of the patent number and a split point is used to separate the sorted patents into two parts. The split point ensures that the number of positive examples is approximately equally in training set and test set. The exact number of training examples and text examples are different for different class, as shown in Table 7-5.

The proposed similarity score is easy to be embedded into a *k*-Nearest Neighbor (*k*-NN) classifier (Manning, Raghavan & Schütze, 2008) by simply replacing the original similarity score. The *k*-NN classifier assigns a class label to an example in test set according to the label(s) of the example's *k* nearest neighbors in training set. The rationale of *k*-NN classification is that, with the

contiguity hypothesis, it is expected that a test example has the same label as the training examples located in the local region surrounding the test example.

Table 7-5 Ten classes and the arrangement of training set and test set

Class	#	# positive training	# positive test	# training	positive rate training
toothbrush	93	44	49	112	39.29%
digital camera	87	43	44	195	22.05%
razor	17	8	9	25	32.00%
lighter	17	7	10	75	9.33%
forceps	13	6	7	13	46.15%
file folder	12	6	6	52	11.54%
mobile phone	12	6	6	195	3.08%
surgical scalpel	11	4	7	38	10.53%
hypodermic needle	6	3	3	33	9.09%
paper punch	5	2	3	34	5.88%

7.4.2 Experimental Results

In the experiments, the proposed iterative similarity scoring did stop before the maximum number of runs i.e., 1000. The distribution of running times is shown in Figure 7-2, where one epoch is defined as one time of iterative computation. As shown in Figure 7-2, most run was stopped after the first epoch. That is because a lack of semantic similarity between two graphs. If a run did not stop after the first epoch, then it usually needs more than ten epochs to stop. The running time is shown in Figure 7-3. Most running time is less than one millisecond. If not, it would take about 17 milliseconds.

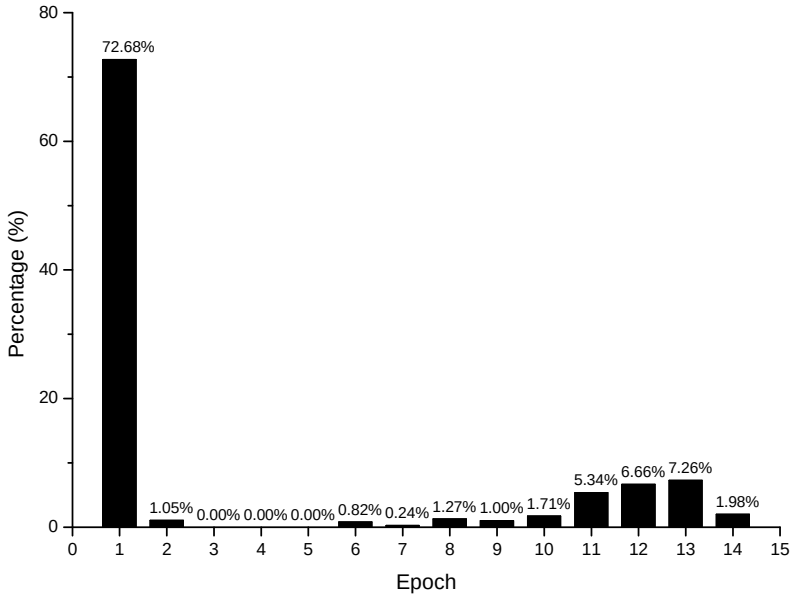


Figure 7-2 The distribution of running epoch of iterative graph similarity scoring

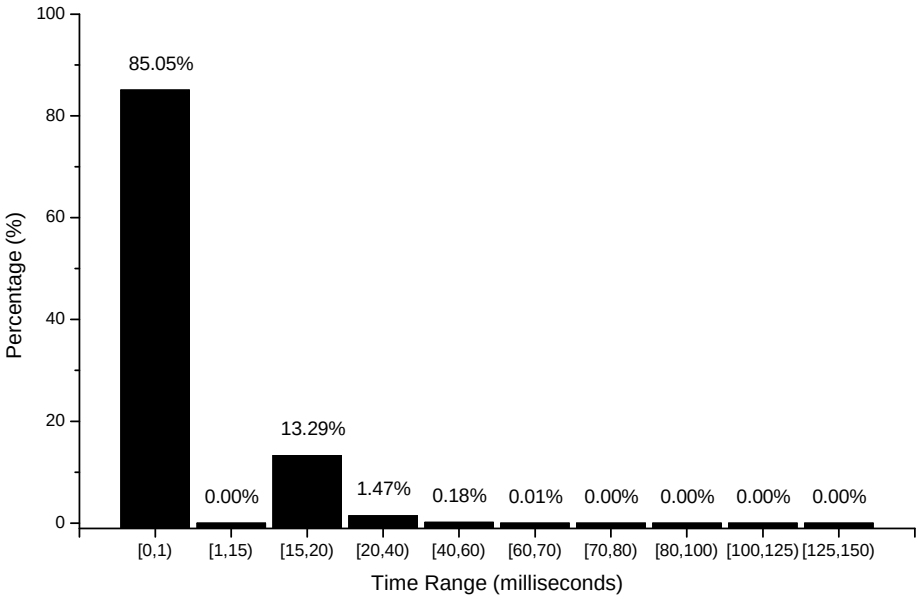


Figure 7-3 The distribution of running time of iterative graph similarity scoring

The proposed classifier was compared with standard *k*-NN classifier and Support Vector Machine (SVM) classifier. The text digitalization adopted a traditional way i.e., extracting title and abstract section and following the preprocessing steps in (Wang, Lu & Loh, 2011). Since both *k*-NN classifier and Support Vector Machine (SVM) classifier have parameters to tune. Parameter tuning was executed and best parameters were used in the method comparison. The F_1 score was used as the evaluation measure.

The *k*-NN classifier can adopt different similarity score. Both cosine similarity and Euclidean distance similarity was test. The *k*-NN classifier got a poor F_1 score when cosine similarity was used. As shown in Figure 7-4, no matter *k* values at $k = 1$, 3 or 5, the F_1 score keeps being zero for eight classes. For the other two classes, the best F_1 score is less than 6%. Therefore, only Euclidean distance was used in the method comparison.

For SVM classifier, the cost parameter *C* was tuned for {0.5, 1, 2, 4, 5, 8, 10, 15, 20}. The results are shown in Figure 7-5. For any class, the F_1 score is stable in the highest value when the cost parameter *C* was set as a high value. Therefore, $C = 20$ was used in the method comparison.

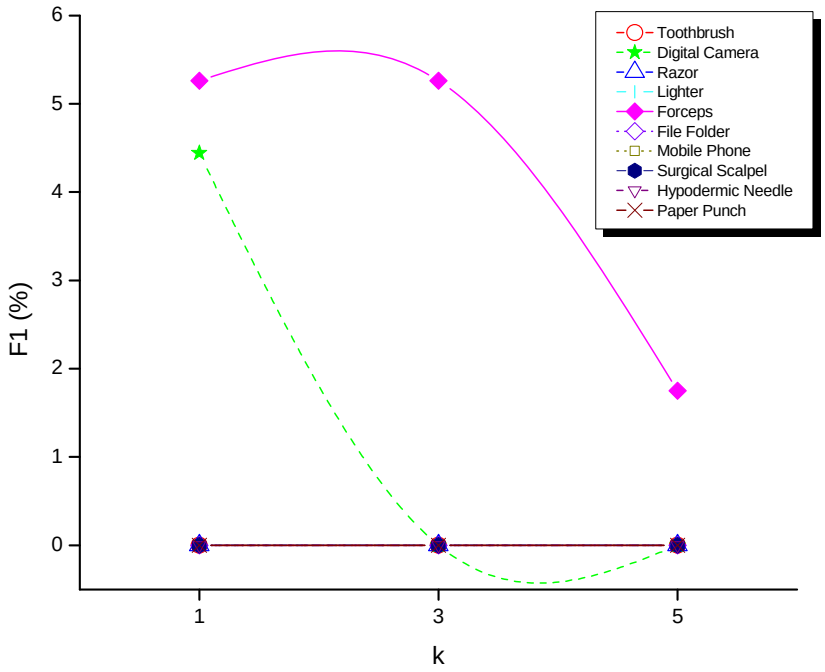


Figure 7-4 The *k*-NN with cosine similarity. Score reported is F_1 measure.

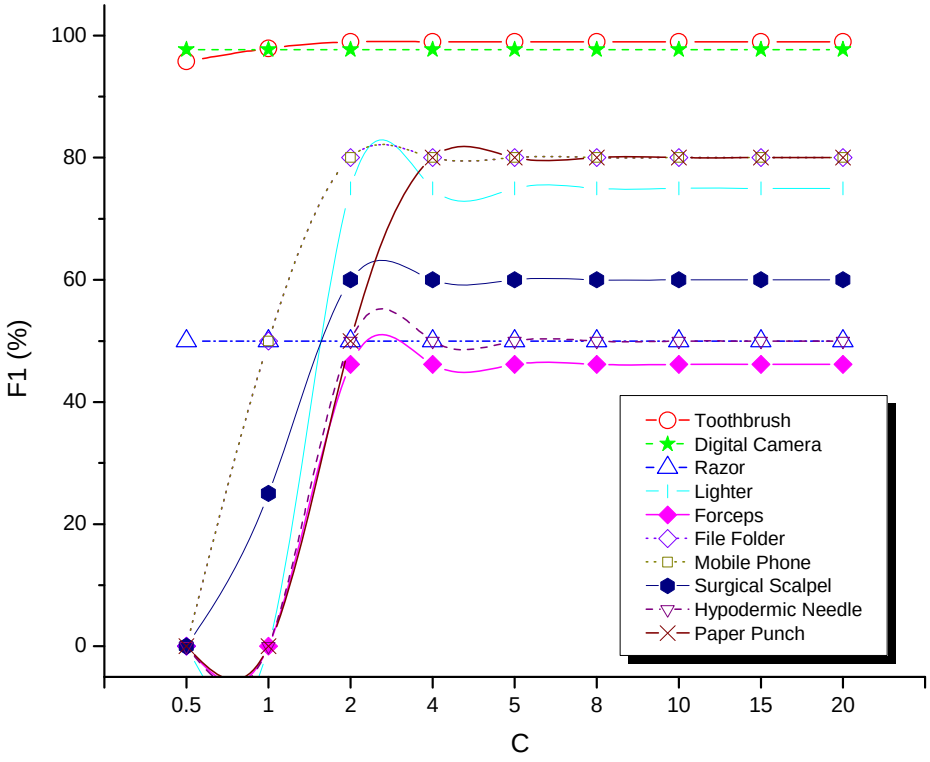


Figure 7-5 The SVM with different *C*. Score reported is F_1 measure.

The results of the method comparison are shown in Figure 7-6. In most cases, *k*-NN with graph similarity could achieve better F_1 score than SVM or standard *k*-NN. However, for class "razor" and class "forceps", *k*-NN with graph similarity

did not perform well. In both cases, the recall is high (more than 85%), but the precision is low (less than 20%). It implies that many negative examples were labeled as positive. Since the *k*-NN classifier predicts class label through the votes of examples in training set that are most close to the test example, a reasonable explanation is a lack of representative negative examples in the training set. In other words, a negative test example is unlike any negative training examples and positive training examples. Thus, it may be more similar to some positive examples, compared to negative examples. In Table 7-5, the class "forceps" has the smallest training set but at the same time has the highest positive rate, while the class "razor" has the second smallest training set but has the third highest positive rate. These facts do not refute above explanation.

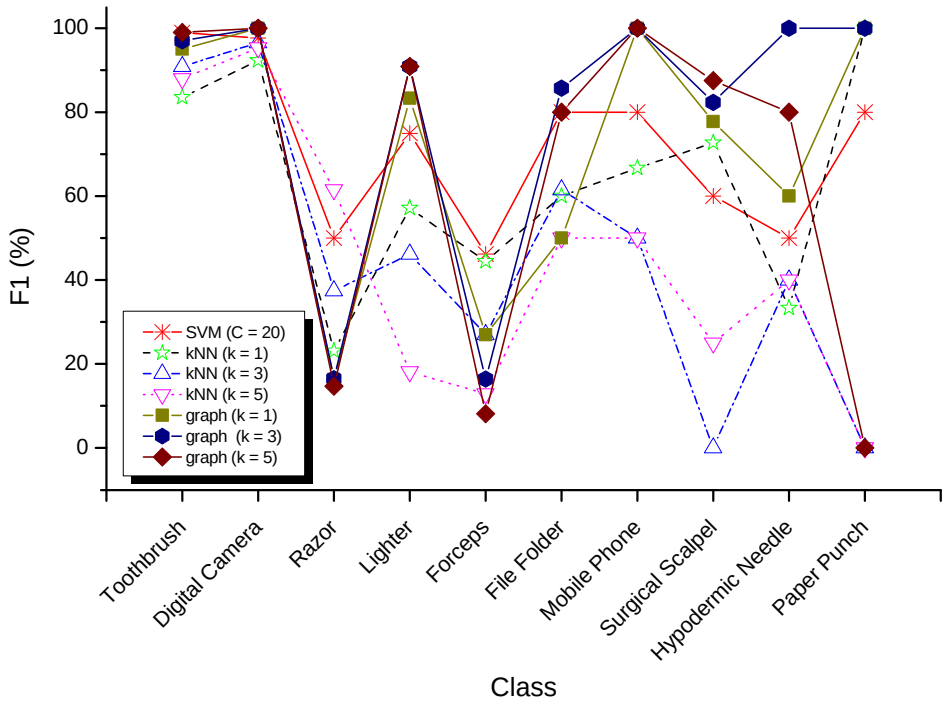


Figure 7-6 Method Comparison: SVM, *k*-NN, and *k*-NN with graph similarity. Score reported is F_1 measure.

To further verify the surmise above, the average similarity of true negative (ASTN) were investigated. The average similarity of true negative is used to evaluate the level at which a true negative is accepted as negative. Given n true negative, the ASTN is defined as below.

$$ASTN = \frac{\sum_{i=1}^{n} \sum_{j}^{k} Similarity(i,j) \Big/ k}{n} \quad (7.8)$$

The ASTN is the average of all similarity scores, each of which is between a negative example and its nearest neighbors. For example, ASTN (k = 3) means three nearest neighbors of each true negative example are considered in ASTN calculation.

As shown in Figure 7-7, it is natural that ASTN (k = 1) > ASTN (k = 3) > ASTN (k = 5), regardless the class. The ASTN of class "razor" has the minimum value. If the class "razor" is excluded, then the ASTN of class "forceps" has the minimum value. It means that negative test examples were less like negative training examples in class "razor" and class "forceps", compared to other eight classes. Therefore, the surmise above is valid.

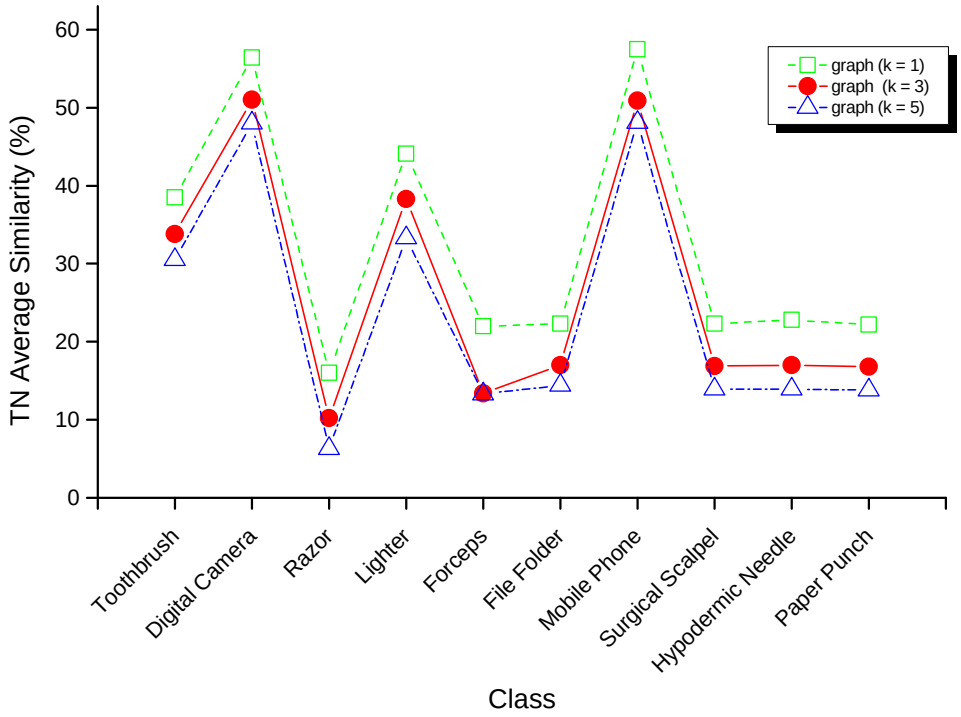


Figure 7-7 The average similarity of true negative

7.5 Summary

To summarize, this chapter proposed similarity measures for the S-model. The proposed similarity measure was presented with a set of graph examples and was tested through a classification problem. The proposed graph similarity demonstrated its superiority in the classification problem. A *k*-NN classifier with the proposed graph similarity measure usually can perform better than a standard *k*-NN classifier or a SVM classifier. However, the performance of the proposed method is sensitive to the representativeness of the training set since it requires a similarity computation between examples.

CHAPTER 8

PATENT GROWTH MAPPER

This chapter introduces the patent growth mapper, discussed in Chapter 3 in detail from the lowest module to the highest module. Since claim dependency parsing for S-model extraction and similarity measures is stated in Chapter 6 and Chapter 7, respectively, this chapter covers the network for clustering and the two-dimensional coordinate system. A case study is given to show the effectiveness of the proposed patent growth mapper.

8.1 Network for Clustering

To monitoring the structural changes of multiple technologies, a network is designed to cluster technologies based on their structure similarity. The network is a graph, in which each node denotes a patented technology (or patent) and similar nodes are linked with edges. A threshold value is used to convert the similarity score between two nodes into binary value i.e., similar or dissimilar.

The network has four characteristics. Firstly, members in each group have similar stricture and likely infringe each other. Secondly, a controllable parameter called connectivity rate is used to adjust the network. The connectivity rate is the fraction of the nodes that are connected. With the connectivity rate, the threshold value does not need to be pre-defined. The bigger the connectivity rate is, the smaller the threshold value is. Different from the threshold value, the connectivity rate is a relative measure of the connectivity of the patent set. Since different technology types usually have different complexity degrees, the similarity distribution of different technology types should be different. A single standard to determining the threshold value does not exist. Therefore, a relative measure is more robust. Thirdly, the number of edges is minimized. The number of edges equals the number of nodes minus one. Fourthly, the size of every group either grows or keeps unchanged. That is why this patent map is called as patent "growth" map.

To control the connectivity rate, trial and error method is used. The algorithm requires a target connectivity rate, an initial threshold value and a step-length. The

target connectivity rate may be at a mid-range value. A high value of connectivity rate reduces discrimination and easily produces big groups, while a low value of connectivity rate leads to a mess. By default, the initial threshold value equals to 0.5, and the step-length equals 0.005. The algorithm firstly calculates the connectivity rate using the initial threshold. When the connectivity rate is bigger than the target connectivity rate, the threshold value is increased regularly according to the step-length, until the connectivity rate is not bigger than the target connectivity rate. A similar process is executed when connectivity rate is smaller than target connectivity rate. Thus, the threshold value of similarity is automatically calculated via the target connectivity rate. A more complex but advanced algorithm may use a binary search tree. However, the proposed algorithm is simple and efficient.

To obtain the minimized number of edges, the clustering algorithm processes node one by one and link node in the process with only one node in every similar group. Formally, a set of patents is a sequence $\{p_1, p_2, p_3, \ldots, p_n\}$. The order is determined by the patent number, since the patent numbers are assigned chronologically. A patent group is a subsequence of the sequence of the patent set. A patent group consists of one or more members. The terminator of a patent group is defined as the last item of the patent group sequence. Given a p_i from $\{p_1, p_2, p_3, \ldots, p_n\}$, the similarity between p_i and p_k ($k = 1, 2, \ldots, i - 1$) is calculated. If the similarity score is larger than the threshold value, p_i is connected to the terminator of the patent group that contains p_k.

8.2 Two-dimensional Coordinate System

To monitoring the trends of structural changes, a two-dimensional coordinate system is designed. Similar to Growth-share Matrix, which is a chart that had been created for the Boston Consulting Group in 1968 to help corporations to analyze their business units or product lines, the design allows the map users to observe patents from two controlling aspects and four quadrants.

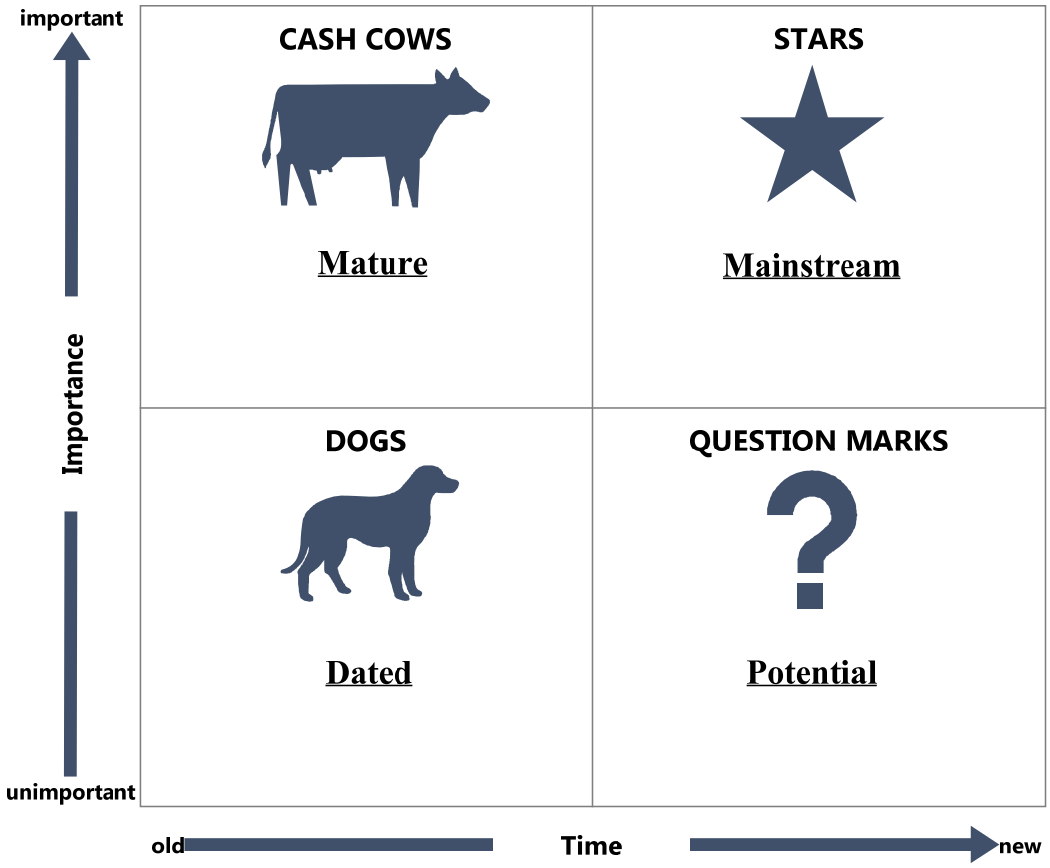


Figure 8-1 The four quadrants of the patent growth map

The two dimensions are timeline (X-axis) and importance (Y-axis). The four quadrants are defined in Figure 8-1. When a technology is new and important, it is considered as mainstream technology; when a technology is new but unimportant, it is considered as potential technology; when a technology is old but important, it is considered as mature technology; when a technology is old and unimportant, it is considered as dated technology.

The time axis (X-axis) is used to demonstrate the trend of technology development. The unit of the time axis is usually year. Normalization is used to convert a year into a value ranging from zero to one.

The importance is designed to highlight new technology and big technology group. The newer the technology is, the more important the technology is. The bigger the technology group is, the more important each technology in the technology group is. Formally, the importance is defined as follows:

$$\text{Importance} \equiv \frac{k}{n} \cdot \frac{\text{Size}(\text{Group}(p_k))}{\max_i \text{Size}(\text{Group}(p_i))} \tag{8.1}$$

Here, k is the sequential number of a patent in the sequence of the patent set; n is the size of the patent set. The function $\mathsf{Group}: p \mapsto g \in G$ returns the group to which the patent p belongs; $\mathsf{Size}: g \mapsto z \in \mathbb{Z}$ returns the size of the group g. The importance score ranges from zero to one.

8.3 Core Technology Selection

To identify core technology, enterprises can select a big technology group in any quadrant according to their strategies. For example, a competitive enterprise may prefer mainstream technology; a risk-averse enterprise may prefer mature technology; a risk-like enterprise may prefer potential technology.

To facilitate identifying the core technology, the core technology is each technology group is automatically selected as the most representative member. The most representative member is defined as the member that is mostly similar to all of the other members in the group. If multiple candidate representative members exist, one of them is selected as the representative member. Formally, the representative member in a group is defined as follows:

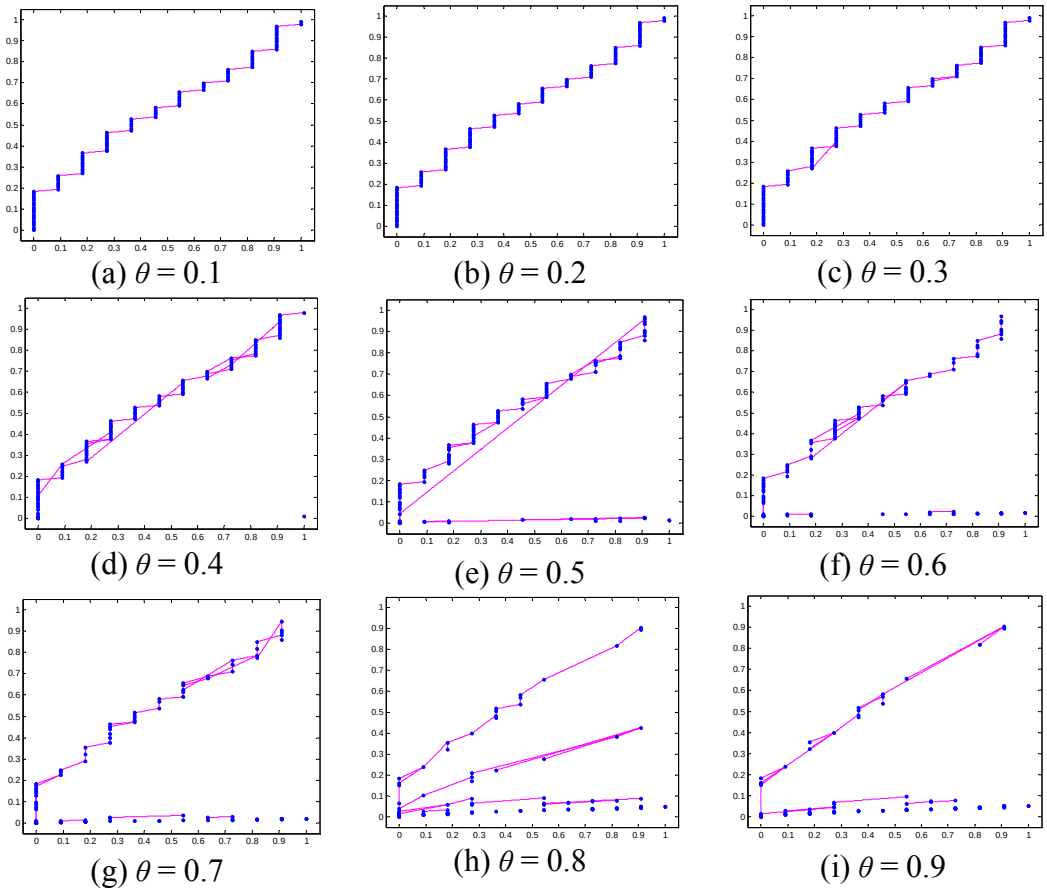
$$i^* = \arg\max_i \sum_j \text{Similarity}(i, j) \tag{8.2}$$

Briefly, the PGM clusters technologies into different groups and distinguishes groups in terms of their positions in the four quadrants. In this way, the designer could target a group of technologies easily. Furthermore, for each technology group, the most representative technology is found. This technology can be directly considered as the core technology.

8.4 Case Study: Patent Growth Map

For generating the PGM, 93 patents of toothbrush were collected from PPAT273. The tree models were extracted from claims and issued years were also extracted.

The patent growth maps of the 93 toothbrushes with different thresholds at {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9} are shown from Figure 8-2. When the threshold is increasing, the dots are more and more scattered. Multiple big groups are visible when threshold value $\theta = 0.8$ as shown in Figure 8-2 (h), in which two important product groups are distinguished. The connectivity rates corresponding to the thresholds are listed in Table 8-1. It is observed that the connectivity rate is 53.76% when threshold θ equals to 0.8. Moreover, it is the one closest to 50% among all thresholds. Therefore, Figure 8-2 (h) is selected for further analysis.



(a) $\theta = 0.1$ (b) $\theta = 0.2$ (c) $\theta = 0.3$

(d) $\theta = 0.4$ (e) $\theta = 0.5$ (f) $\theta = 0.6$

(g) $\theta = 0.7$ (h) $\theta = 0.8$ (i) $\theta = 0.9$

Figure 8-2 An example of growth map with θ from 0.1 to 0.9

Table 8-1 The threshold similarity value and corresponding connectivity rate

Threshold θ	Connectivity Rate r_c.
0.1	100%
0.2	100%
0.3	100%
0.4	97.85%
0.5	92.47%
0.6	76.34%
0.7	64.52%
0.8	53.76%
0.9	36.56%

To select core product, product groups are firstly selected. As shown in Figure 8-3 (when θ = 0.8), there are two most important product groups: Group 1 and Group 8. Group 1 has 20 members, some of which are considered as mainstream products according to defined four quadrants. Group 8 has nine members, some of which are considered as potential products according to defined four quadrants.

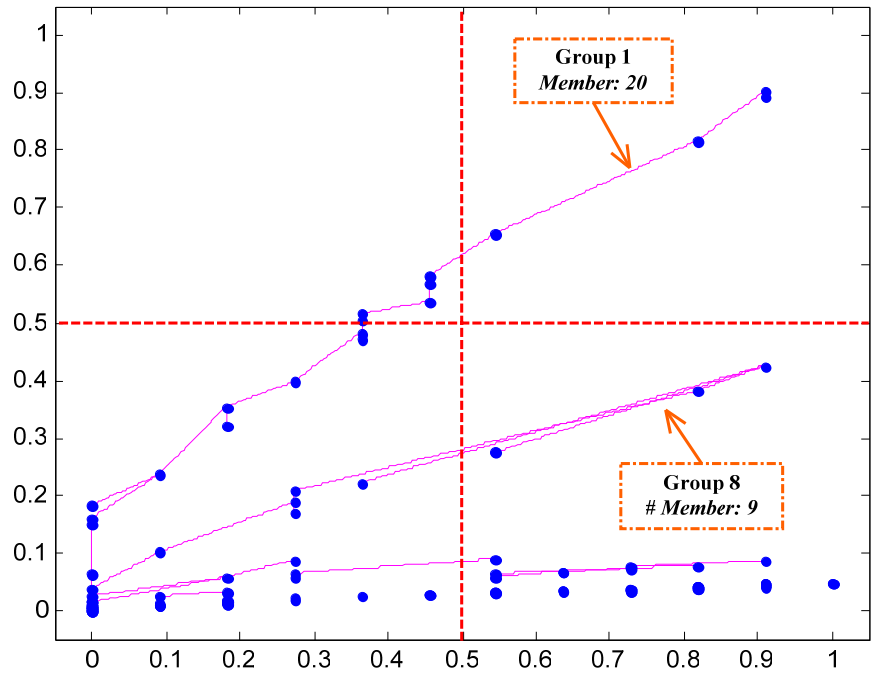


Figure 8-3 An example of growth map with θ = 0.8, where two most important groups are highlighted

It was observed that the structure of toothbrushes in Group 1 is simple. The representative toothbrush (in patent numbered US6115870) simply comprises of head, handle and bristles. The field of product is the bristle arrangement. For toothbrushes in Group 1, other filed of product includes bendable head, polishing

element, and aesthetic design. This conclusion is consistent with previous work (Hohlbein, Williams & Mintel 2004) supported by the Colgate-Palmolive company. It said that the trend of toothbrush development is to consider new material and product esthetics. Therefore, mainstream toothbrush may follow simple structure, but improve look and properties with new material. Such improvement has little impact on consumer's use habits.

In contrast, the structure of toothbrushes in Group 8 is complex. The thread of design is to involve some novel parts. For example, the representative toothbrush (in patent numbered US6308367) is about a toothbrush with a three-dimensional bristle profile to provide improved cleaning of interproximal and gingival marginal regions of teeth. Such improvement may change consumer's use habits. Therefore, enterprises should be very careful when using these potential technologies.

8.5 Summary

This paper proposed PGM for monitoring trends of technological changes via measuring structural changes of patented products. In this way, the trends of technological changes can be observed and core products are easy to target.

The PGM organizes a set of patents into a two-dimensional patent map and is them into different groups. The two-dimensional coordinate system distinguishes groups with four quadrants. The PGM users may select different groups according to different strategies. The groups are easy to see, since the number of edges is minimized. To facilitate avoidance of patent infringement, each group consists of structure-similar patented technologies. Furthermore, core patent is automatically highlighted. With the PGM, product designers can observe technological development easily and target core products easily. Moreover, with technology comparison capability and the detailed structure of technology, the scope of the prior art is much clearer. Thus, designers can obtain a boarder and more detailed view on prior art and a correct judgment on their own innovation.

The PGM is an efficient tool, which is able to automatically compare a large number of similar technologies. In this way, product designers are able to grasp hundreds of patents or thousands of patent claims in minutes. Thus, the product

designers obtain a capability that was hitherto impossible and allows them to finish their work in a shorter time.

CHAPTER 9

CONCLUSIONS AND RECOMMENDATIONS

This chapter gives a final evaluation of the hypothesis of this thesis. It also summarizes major discoveries and contributions. Finally, it gives recommendations for future work.

9.1 Final Evaluation of the Hypothesis

The final evaluation of the hypothesis is summarized in Table 9-1.

Table 9-1 The final evaluation of the hypothesis

Objectives	Evaluation	Details
Extract automatically E-model	Partially achieved.	Effects can be extracted with a query-focused dependency-parsing-based method.
Extract automatically S-model	Partially achieved.	A new parser is proposed. Although perfect S-model extraction cannot be achieved with the proposer parser, it is efficient and much better than the state of art.
Compare S-models	Achieved.	A new graph similarity measure is proposed and evaluated.
Improve patent search with E-model	Achieved.	An effect-oriented search engine is proposed. Those patents that do not contain queried effect have lowly ranked and can be filtered out.
Improve patent clustering with S-model	Achieved.	A patent growth map is proposed. Each cluster consists of technologies that likely infringe each other.
Hypothesis	Partially achieved	

9.2 Contributions

New knowledge obtained and the difference between the new knowledge and the state of art is summarized in Table 9-2. Briefly, this thesis proposes technology ontology and a framework to utilize the technology ontology in patent information access. Any technology is characterized by its effect (modeled as a triple i.e., E-model) and its structure (modeled as a tree i.e., S-model).

Table 9-2 The summary of contributions

Contributions	Advance	State of Art
A new entity recognition method	Relatively good (in terms of F_1 measure)	Other participants in NTCIR-8 (2010)
An effect-oriented patent search engine New Features: (1) Effect-oriented (1) Cross-POS expansion (2) Morphology expansion: Inflection	Effect information can be used as a filter to reduce the number of returned patents. Both syntactic and semantic search.	Google Patent Search Engine (or other search engines based on standard Boolean model) Goldfire (semantic search)
A new dependency parsing method for patent claims	Obviously improvement in S-model extraction (in terms of accurate rate and parsing time)	Stanford parser
Two Graph Similarity Measures The latter is recommended. (1) Weighted node-to-node scoring (2) Iterative node-to-node scoring	Relatively good in patent classification (in terms of F_1 measure) Handling edge similarity appropriately; Keeping initial relative semantic similarity	VSM based similarity measures Iterative graph similarity measure (Zager &Verghese, 2008)
Patent Growth Map (PGM) New Features: (1) Technologies in the same cluster are similar in structure and are likely to infringe each other. (2) Each patent is represented as S-Model rather than VSM. (3) Network with controllable connectivity rate and minimized edge number (4) Coordinate system showing trend and facilitating selection of core technology	Considering patent infringement in clustering; Other designs for ease of use	Patent Map via VSM (Lee, Yoon & Park, 2009; Tseng, Lin & Lin, 2007) Patent Map via VSM with Network (Yoon & Park, 2004)

To extract E-model, a new entity recognition method is proposed. The method was evaluated in a cutting edge patent information access evaluation, in which the

NER that focus on technology entities and effect entities was investigated in a large-scale for the first time. The method was the number one according to the evaluation results.

To utilize the extracted E-models, an effect-oriented patent search engine is introduced. Compared to traditional search engine, it uses effect information as a filter to reduce the number of returned patents. Both syntactic and semantic technologies are used.

To extract S-model, the Independent Claim Segment Dependency Syntax (ICSDS) was proposed for parsing claims. Although perfect S-model extraction cannot be achieved with the proposer parser, it is efficient and much better than the state of art in terms of accurate rate.

To compare technologies, new graph similarity measures were proposed. The recommended graph similarity measure shows its superiority in a classification problem. However, the performance of proposed method is sensitive to the representativeness of the training set, since it requires similarity computation between two examples.

To utilize the extracted S-models and recommended graph similarity measure, a new patent map i.e., PGM was proposed. In the PGM, technologies that likely infringe each other are grouped together. With the growth map, product designers can target core technologies easily.

The proposed methods promote the processing of patent information in a deeper, larger, and faster way. At the same time, they promote the reduction of human effort on reading patent documents and gathering information. A designer can obtain a capability that was hitherto impossible and have a boarder and more detailed view on prior art and a correct judgment on his own innovation. Moreover, they will have more time to focus on creative work.

9.3 Recommendations for Future Work

(1) Extracting correct technology

For simplification, the technology TechnologyName in the E-model (TechnologyName, PropertyName, PropertyChange) is assumed to be known (see

Chapter 5). To obtain more precise relation, the correct technology i.e., the agent of the effect is necessary to be identified. The TechnologyName may be a set of technology, if the effect is caused by several technologies. Apart from syntactical analysis, coreference resolution analysis is also required.

(2) Expanding the ICSDS by defining more relationships between segments

The current implementation of ICSDS focuses on verb-noun relation and adjective-noun relation (see Chapter 6). This is because they are the most important relations for effect discovery and are difficult to correctly parse. However, for completeness, other relations such as preposition-noun, verb-preposition and adverb-verb should also be defined. Therefore, relationships between segments are worth further studying.

(3) Considering more patterns of effect expression

Some patterns of effect expression, including negator and adverb (see Appendix I), have not been implemented. Additional work is required to enable the use of negator and adverbs. A negator or an adverb usually works as a modifier of the center word. They can work separately or collectively to change the semantics.

Besides, the discussed patterns applicable to text did not consider numerals. In the future, more patterns can be designed to include numerals.

(4) Product concept design module

In the proposed framework, it is expected that the proposed technology ontology can support product concept design and development. Especially, the technology ontology is expected to facilitate designing around multiple existing patents. A systematic methodology has not been proposed yet. The systematic methodology may require some new intelligent technologies, for example automated generation of patentable candidate product concept model.

(5) Other text-based applications

In the knowledge discovery module of the proposed framework, only the patent classification was investigated. Other applications like patent summarization or question-answering can also be explored.

(6) Integrated patent search and analysis platform

The terminal carrier of all proposed technologies will be an integrated patent search and analysis platform. Since current trend of information technology is towards high performance computing and wireless connection, the terminal platform should be a cloud computing platform. More works are needed to realize such platform.

BIBLIOGRAPHY

Agichtein, E. & L. Gravano (2000). Snowball: Extracting Relations from Large Plain-text Collections. In *Proceedings of DL'00, the 5th ACM Conference on Digital Libraries*

Ahmad K. & L. Gillam (2005). Automated Ontology Extraction from Unstructured Texts. In Meersman R. & Tari Z. (Eds.), *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, LNCS 3761* (pp. 1330-1346). Berlin Heidelberg: Springer-Verlag

Andreevskaia, A. and S. Bergler (2006). Mining WordNet for A Fuzzy Sentiment: Sentiment Tag Extraction from WordNet Glosses. 11th Conference of the European Chapter of the Association for Computational Linguistics.

Appelt, D. & D. Israel (1999). Introduction to Information Extraction Technology. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, Stockholm, Sweden.

Banko, M., M. Cafarella & S. Soderland et al. (2007). Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. Hyderabad, India.

Benz D. (2007). Collaborative ontology learning. Master's Thesis, University of Freiburg

Bezdek, J. C. (1981). Pattern Recognition with Fuzzy Objective Function Algoritms, Plenum Press, New York.

Borst P. & H. Akkermans (1997). An Ontology Approach to Product Disassembly. In *Proceedings of EKAW'97, the 10th European Workshop on Knowledge Acquisition, Modeling and Management*

Bratus S., A. Rumshisky & A. Khrabrov et al. (2011). International Journal on Document Analysis and Recognition – Special Issue on Noisy Text Analytics, 14(2).

Briggs, T., B. Iyer & P. Carlile. (2007). The Co-evolution of Design and User Requirements in Knowledge Management Systems: the Case of Patent

Management Systems. In *Proceedings of HICSS'07, 40th Hawaii International Conference on System Sciences*

Brin S. (1998). Extracting Patterns and Relations from the World Wide Web. In *Proceedings of WebDB'98, the International Workshop on the World Wide Web and Databases*

Bunescu R. & R. Mooney (2005). A Shortest Path Dependency Kernel for Relation Extraction. In *Proceedings of HLT'05, the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*

Cavallucci, D. & N. Khomenko (2007). From TRIZ to OTSMTRIZ: Addressing Complexity Challenges in Inventive Design. International Journal of Product Development, 4(1-2), 4-21.

Cer D., M.-C. Marneffe & D. Jurafsky et al. (2010). Parsing to Stanford Dependencies: Trade-offs between Speed and Accurary. In *Proceedings of LREC 2010, the 7th International Conference on Language Resources and Evaluation*

Choudhary, B. and P. Bhattacharyya (2002). Text Clustering using Semantics. 11th International World Wide Web Conference.

Deerwester, S., S. T. Dumais & G. W. Furnas, et al. (1990). Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science.

Dunn, J. C. (1973). "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters." Cybernetics and Systems 3(3): 32-57.

Eichler, K., H. Hemsen & G. Neumann (2008). Unsupervised Relation Extraction from Web Documents. In *Proceedings of the 6th Edition of the Language Resources and Evaluation Conference*

Eisner., J. (1996) Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of COLING*

Engler, J. & A. Kusiak (2008). Web Mining for Innovation. ASME Mechanical Engineering, 130(11), 38-40.

Etzioni, O., M. Cafarella & D. Downey et al. (2005). Unsupervised Named-entity Extraction from the Web: an Experimental Study. Artificial Intelligence, 165(1)

Fellbaum C. (1998). WordNet: an Electronic Lexical Database, MIT Press, Cambridge, MA

Fujii, A., M. Iwayama & N. Kando (2004) Overview of Patent Retrieval Task at NTCIR-4. In *Proceedings of NTCIR-4*, Tokyo

Gaeta M., F. Orciuoli & S. Paolozzi et al. (2011). Ontology Extraction for Knowledge Reuse: the E-learning Perspective. IEEE Transaction on Systems, Man and Cybernetics – Part A: Systems and Humans, 41(4)

Gero J. S. & U. Kannengiesser (2003). A Function-behaviour-structure View of Socially Situated Design Agents. In *Proceedings of the CAADRIA03*

Ghoula N., K. Khelif & R. Dieng-Kuntz (2007). Supporting Patent Mining by using Ontology-based Semantic Annotations. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Web Intelligence*

Giereth M., S. Koch & Y. Kompatsiaris et al. (2007). A Modular Framework for Ontology-based Representation of Patent Information. In *Proceedings of JURIX 2007, the 2007 Conference on Legal Knowledge and Information Systems*.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2), 199-220.

Han Y. & Y. Park (2006). Patent Network Analysis of Inter-industrial Knowledge Flows: the Case of Korea between Traditional and Emerging Industries. World Patent Information, 28(3), 235-247.

Hearst, M. A. (1999). Untangling Text Data Mining. In Proceedings of ACL'99, the 37th Annual Meeting of the Association for Computational Linguistics, invited paper, oxford university press, 2003.

Hofmann, T. (1999). Probabilistic Latent Semantic Indexing. 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Berkley, CA, US.

Hohlbein D. J., M. I. Williams & T. E. Mintel (2004). Driving Toothbrush Innovation through a Cross-functional Development Team. Compendium of Continuing Education in Dentistry, 25(10), (supplement 2), 7-11.

Hotho, A., S. Staab, & G. Stumme (2003a). Explaining Text Clustering Results Using Semantic Structures. Knowledge Discovery in Databases: PKDD 2003.

Hotho, A., S. Staab, & G. Stumme (2003b). Ontologies Improve Text Document Clustering. 3rd IEEE International Conference on Data Mining: 541-544.

Hotho, A., S. Staab, & G. Stumme (2003c). Text Clustering Based on Background Knowledge, Institute AIFB, University of Karlsruhe.

Hung Y. & Y. Hsu (2007). An Integrated Process for Designing around Existing Patents through the Theory of Inventive Problem-solving. Proceedings of the Institution of Mechanical Engineers Part B: Journal of Engineering Manufacture, 221(1), 109-122

Hunt, D., L. Nguyen & M. Rodgers (2007). *Patent Searching: Tools & Techniques*. John Wiley & Sons

Hwang, C. H., B. W. Miller & M. E. Rusinkiewicz (2002). *Ontological Concept-based, User-centric Text Summarization*. United States Patent Application Publication

Ide, N. & J. Veronis (1998). Introduction to the Special Issue on Word Sense Disambiguation: the State of the Art. Computational Linguistics 24(1).

Jiang J. & C. Zhai (2007). A Systematic Exploration of the Feature Space for Relation Extraction. In *Proceedings of the NAACL-HLT'07: Human Language Technologies: the Conference of the North American Chapter of the Association for Computational Linguistics*

Jouili, S., S. Tabbone & E. Valveny (2010). Comparing Graph Similarity Measures for Graphical Recognition. In Ogier J.-M. et al. (Eds.), *Graphics Recognition, Achievements, Challenges, and Evolution, Lecture Notes in Computer Science, Volume 6020* (pp. 37-48). Berlin Heidelberg: Springer-Verlag

Kambhatla, N. (2004). Combining Lexical, Syntactic, and Semantic Features with Maximum Entropy Models for Extracting Relations. In *Proceedings of the ACLdemo'04: the ACL 2004 on Interactive Poster and Demonstration Sessions*, Stroudsburg, PA, USA

Kato, T. & M. Matsushita. (2008) Overview of MuST at the NTCIR-7 Workshop: Challenges to Multi-model Summarization for Trend Information. In *Proceedings of NTCIR-7*, Tokyo, Japan

Kleinberg, J. M. (1999) Authoritative Sources in a Hyperlinked Environment. Journal of the ACM, 46, 614-632

Kuhn, H. (1955) The Hungarian Method for the Assignment Problem. Naval Research Logistic Quarterly, 2, 83-97

Kushmerick, N., D. S. Weld & R. Doorenbos (1997). Wrapper Induction for Information Extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Aichi, Japan.

Lafferty, J., A. McCallum & F. C. N. Pereira (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the International Conference on Machine Learning*.

Lee C., J. Jeon & Y. Park (2011). Monitoring Trends of Technological Changes Based on the Dynamic Patent Lattice: a Modified Formal Concept Analysis Approach. Technol. Forecast. Soc. Change, 78, 690-702.

Lee S., B. Yoon & Y. Park (2009). An Approach to Discovering New Technology Opportunities: Keyword-based Patent Map Approach, Technovation 29, 481-497.

Li, M. (2011). Similarity Assessment and Retrieval of CAD Models. PhD's Thesis, National University of Singapore

Lin D. C., J. Liou, J. Du, C. H. Lin, S. W. Tu, H. Y. Tseng, C. Y. Chen, Y. C. Lee (2005). Automatic Patent Claim Reader and Computer-aided Claim Reading Method. United States Patent Application Publication, US 2005/0004806 A1

Liu, C.-Y. & S.-Y. Luo (2007). Applying Patent Information to Tracking a Specific Technology. Data Science Journal, 6

MacQueen, J. B. (1967). Some Methods for Classification and Analysis of Multivariate Observations. 5th Berkeley Symposium on Mathematical Statistics and Probability.

Maedche, A. & S. Staab (2001). Ontology Learning for the Semantic Web. IEEE Intelligent Systems, 16(2)

Manning, C.D., P. Raghavan & H. Schütze (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Marneffe, M., B. MacCartney & C. D. Manning (2006). Generating Typed Dependency Parses from Phrase Structure Parses. In *LREC 2006*.

Martino Joseph P. (1993). *Technological Forecasting for Decision Making*. McGraw-Hill

Miller, G. A. (1995). WordNet: A Lexical Database for English. Communications of the ACM, 38(11), 39-41

Miller, S., M. Crystal & H. Fox et al. (1998). Algorithms that Learn to Extract Information BBN: Description of the Sift System as Used for MUC-7. In *Proceedings of the MUC-7: Message Understanding Conference*

Mintz, M., S. Bills & R. Snow et al. (2009). Distant Supervision for Relation Extraction without Labeled Data. In *Proceedings of ACL'09, the Joint Conference of the 47th Annual Meetings of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Stroudsburg, PA, USA

Nadeau, D. & S. Sekine (2007). A Survey of Named Entity Recognition and Classification. Lingvisticae Investigationes, 30(1), 3-26

Nivre J. (2005). Dependency Grammar and Dependency Parsing. Technical Report. Växjö University

Nivre J. & R. McDonald (2008) Integrating Graph-based and Transition-based Dependency Parsers. In *Processing of ACL-HLT*

Nivre J. & M. Scholz (2004) Deterministic Dependency Parsing of English Text. In *Processing of COLING*

Oluikpe, P., P. M. Carrillo, & J. A. Harding, et al. (2008). Text Mining of Post Project Reviews. Performance and Knowledge Management.

Osada, R., T. Funkhouser & B. Chazelle et al. (2002) Shape Distributions. ACM Transactions on Graphics, 21(4), 807-832

OuYang, H. & C. S. Weng (2011) A New Comprehensive Patent Analysis Approach for New Product Design in Mechanical Engineering. Technological Forecasting & Social Change, 78, 1183-1199

Parapatics P. & M. Dittenbach (2011). Patent Claim Decomposition for Improved Information Extraction. In Lupu M. et al. (Eds.), *Current Challenges in Patent Information Retrieval* (pp. 197-216). Berlin Heidelberg: Springer-Verlag

Rosenfeld, B., R. Feldman & M. Fresko et al. (2006) TEG – A Hybrid Approach to Information Extraction. Knowledge and Information Systems, 9, 1-18.

Russo, D. (2010). Knowledge Extraction from Patent: Achievements and Open Problems: A Multidisciplinary Approach to Find Functions. In *Proceedings of the 20th CIRP Design Conference*, Nantes, France.

Sagae, K. & A. Lavie (2006) Parser Combination by Reparsing. In *Proceedings of HLT-NAACL*

Sang, E. F. T. K. & F. D. Meulder (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Conference on Natural Language Learning*.

Sarawagi, S. (2007). Information Extraction. Foundations and Trends in Databases, 1(3), 261-377.

Settles, B. (2004). Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets. In *Proceedings of the International Joint workshop on Natural Language Processing in Biomedicine and its Applications*. Geneva, Switzerland.

Sha, F. & F. Pereira (2003). Shallow Parsing with Conditional Random Fields. In *Proceedings of HLT/NAACL*.

Shih, M. & D. Liu (2010). Patent Classification Using Ontology-based Patent Network Analysis. In *Proceedings of PACIS, the Pacific Asia Conference on Information Systems.*

Shinmori, A. & M. Okumura (2004). Can Claim Analysis Contribute toward Patent Map Generation? In *Proceedings of NTCIR-4*. Tokyo.

Shinyama, Y. & S. Sekine (2006). Preemptive Information Extraction Using Unrestricted Relation Discovery In *Proceedings of HLT-NAACL'06, the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics.* Stroudsburg, PA, USA.

Soderland, S. (1999). Learning Information Extraction Rules for Semi-structured and Free Text. Machine Learning 34, 233-272.

Studer R., V. R. Benjamins & D. Fensel (1998) Knowledge Engineering: Principles and Methods. Data and Knowledge Engineering, 25, 161-197

Strzalkowski, T. & B. Vauthey (1992). Information Retrieval Using Robust Natural Language Processing. In *Proceedings of ACL'92, the 30th Annual Meeting on Association for Computational Linguistics*, 104-111

Taduri, S., G. T. Lau & K. H. Law et al. (2011). An Ontology-based Interactive Tool to Search Document in the U.S Patent System. In *Proceedings of the 12th Annual International Conference on Digital Government Research*, College Park, MD, USA

Toutanova, K. & C. D. Manning (2000). Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *Proceedings of EMNLP/VLC-2000 the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, Hong Kong.

Toutanova, K., D. Klein & C. D. Manning et al. (2003). Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL 2003*

Trappey, A. J. C. & C. V. Trappey (2008). An R&D Knowledge Management Method for Patent Document Summarization. Industrial Management & Data Systems 108(2).

Tseng, Y.-H., C.-J. Lin & Y.-I. Lin (2007). Text Mining Techniques for Patent Analysis. Information Processing and Management, 43(5), 1216-1247.

Uchida, H. & A. Mano (2004). Patent Map Generation Using Concept-based Vector Space Model. In *Proceedings of NTCIR-4*, Tokyo.

Ulrich, K. T. & S. D. Eppinger (2008). *Product Design and Development*. Boston: McGraw-Hill/Irwin

Uschold M. & M. Gruninger (1996). Ontologies: Principles Methods and Applications, Knowledge Engineering Review, 11, 93-136

Wallach, H. M. (2004). *Conditional Random Fields: An Introduction. Technical Report*. University of Pennsylvania

Wanner, L., R. Baeza-Yates & S. Brugmann et al. (2008). Towards Content-oriented Patent Document Processing. World Patent Information, 30(1), 21-23.

Wang, J., H.T. Loh & W. F. Lu (2010) Extracting Technology and Effect Entities in Patents and Research Papers. In *Proceedings of NTCIR-8.* Tokyo, Japan

Wang, J., W. F. Lu & H.T. Loh (2011) P-SMOTE: One Oversampling Technique for Class Imbalanced Test Classification. In *Proceedings of IDETC/CIE 2011, ASME 2011 International Design Engineering Technical Conference & Computers and Information in Engineering Conference.* Washington D.C., USA

Wang, Q. I., D. Lin & D. Schuurmans (2007) Simple Training of Dependency Parsers via Structured Boosting. In *Proceedings of IJCAI*

Ward, J. H., Jr. (1963). "Hierarchical Grouping to Optimize an Objective Function." Journal of the American Statistical Association 58(301): 236-244.

Wberry, T. L. (1995). *Patent Searching for Librarians and Inventors*. American Library Association.

Xiao, J., T.-S. Chua, & J. Liu. (2003). A Global Rule Induction Approach to Information Extraction. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*.

Yamada, H. & Y. Matsumoto (2003) Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of the IWPT*

Yang, S.-Y., S.-Y. Lin, & S.-N. Lin et al. (2005). An Ontology-based Multi-agent Platform for Patent Knowledge Management. International Journal of Electronic Business Management 3(3), 181-192.

Yao, B., P. Jiang & T. Zhang et al. (2010). A Study of Designing around Patents Based on Function Trimming. In *Proceedings of ICMIT2010, 5th IEEE International Conference on Management of Innovation and Technology*

Yoon B. & Y. Park (2004). A Text-mining-based Patent Network: Analytical Tool for High-technology Trend. Journal of High Technology Management Research, 15, 37-50.

Zager, L. A. & G. C. Verghese (2008). Graph Similarity Scoring and Matching. Applied Mathematics Letters, 21, 86-94.

Zanasi, A. (2005) Text Mining and its Applications to Intelligence, CRM and Knowledge Management, WIT Press.

Zhang, Y. & S. Clark (2008) A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing Using Beam-search. In *Proceedings of EMNLP*

Zhao, S. & R. Grishman (2005). Extracting Relations with Integrated Information Using Kernel Methods. In *Proceedings of ACL'05, the 43rd Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, US.

Zhi, L. & H. Wang (2009). A Construction Method of Ontology in Patent Domain Based on UML and OWL. In *Proceedings of the International Conference on Information Management, Innovation Management and Industrial Engineering*.

Zhou, C., H. Chen, & J. Tao (2011). GRAPH: a Domain Ontology-driven Semantic Graph Auto Extraction System. Applied Mathematics & Information Sciences, 5(2)

Zhou, G. & J. Su (2001). Named Entity Recognition Using an HMM-based Chunk Tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, Pennsylvania, US.

Zhou, G., J. Su, & J. Zhang et al. (2005). Exploring Various Knowledge in Relation Extraction. In *Proceedings of the ACL'05: the 43rd Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, US.

APPENDIX I

SYNTACTIC PATTERNS FOR EXPRESSING EFFECT

Before listing the discovered syntactic patterns, several symbols are defined in order to describe the syntactic relation:

"◄" means the element on the right is towards the center i.e., the element on the left;

"+" means the element on the right is necessarily added to the element on the left;

"\" means the element on the left having a specific form, which is morphologically related to the element on the right.

It should be noted that the element order in these syntactic patterns does not correspond with the practical token order in natural language. An object element is always put at the beginning of a pattern.

(1) Adjective-like character

An adjective-like character is a descriptor such as an adjective, a noun, or a noun phrase. The adjective may be in its comparative form. No matter its specific type, the descriptor works like an adjective. It modifies an object in one of manners below:

Pattern (object ◄ adjective): *efficient* charging

Pattern (object ◄ adjective + preposition): *high in* sensitivity

Pattern (object ◄ adjective + preposition): *free from* error

Pattern (object ◄ adjective + noun): *high quality* recording

Pattern (object ◄ preposition + adjective\comparative + noun): image *of higher quality*

Pattern (object ◄ adjective + noun + preposition): *small amount of* force

Pattern (object ◄ noun + preposition): *reduction of* cost

Pattern (object ◄ noun): cost *reduction*

Moreover, the adjective may be modified and limited by an adverb.

Pattern (object ◄ adjective ◄ adverb): *highly efficient* charging

Besides, the adjective-like character may rely on a verb and works as a complement or more specifically a predicative.

Pattern (object ◄ linking verb + adjective): The cost is *high*.

Pattern (object ◄ linking verb + preposition + noun phrase): The thickness *is at nanometer level*.

(2) Verb-like behavior

A verb-like behavior must include a verb which is considered as the behavior of the object. The object and the verb constitute a part of a predicate-argument structure, in which the verb is the predicate and the object is an argument, either a subject or a grammatical object. The form of the verb and its position is influenced by the grammatical structure, for example, passive voice, active voice or a syntactic expletive.

Pattern (object ◄ verb\infinitive): *reduce* the cost

Pattern (object ◄ verb\third person singular): *reduces* the cost

Pattern (object ◄ verb\present participle): *reducing* the cost

Pattern (object ◄ auxiliary verb + verb\past participle): the cost *is reduced*

Pattern (object + syntactic expletive ◄ auxiliary verb + verb\past participle): There can be *obtained* the cost.

Sometimes, the verb is attached with a preposition to form a collation.

Pattern (object ◄ auxiliary verb + verb\past participle + preposition): The transistor can *be turned off*.

Moreover, the verb may be modified and limited by an adverb or a preposition phrase.

Pattern (object ◄ verb ◄ adverb): *efficiently improving* the reliability

Pattern (object ◄ verb ◄ adverb): *improving efficiently* the reliability

Pattern (object ◄ auxiliary verb + verb\past participle ◄ preposition phrase): The delay *is cut by half*.

(3) Adjective compound

Adjective compound is composed of an adjective and a noun (or an adverb), through a hyphen. They work in the same manner as that of adjectives.

Pattern (adjective compound): high-quality

Pattern (adjective compound): ever-higher

(4) Negator

A negator may be added to reverse the semantics.

Pattern (object ◄ negator) *no* cost

Pattern (object ◄ negator): *without* picture disruption

Pattern (object ◄ linking verb + adjective ◄ negator): The cost *is not high*.

Pattern (object ◄ verb ◄ negator): *without reducing* the reliability

Pattern (object ◄ auxiliary verb + verb\past participle ◄ negator): Transition *is not required*.

It was observed that the use of negator is very flexible. The negator can be used together with noun, adjective and verb.

APPENDIX II

TYPES OF SEQUENTIAL NUMBER

There are five types of sequential number in independent claim.

Type A: a sequential Roman number enclosed with a pair of round brackets or parentheses i.e. “(” and “)”. **Examples:** (i), (ii), (iii), (iv)

Type B: a sequential Roman number followed with a closing round brackets or parentheses “)”. **Examples:** i), ii), iii), iv)

Type C: an alphabetical sequential number enclosed with a pair of round brackets or parentheses i.e. “(” and “)”. **Examples:** (a), (b), (c), (d)

Type D: an alphabetical sequential number followed with a closing round brackets or parentheses “)”. **Examples:** a), b), c), d)

Type E: an alphabetical sequential number followed with a period “.”. **Examples:** a., b., c., d.