Category Tree Integration by Exploiting Hierarchical Structure

LIN, Jianfeng

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Systems Engineering and Engineering Management

September 2007

Thesis/Assessment Committee

Professor Lam, Wai (Chair)
Professor Yang, Chuen-Chi Christopher (Thesis Supervisor)
Professor Meng, Mei-Ling Helen (Committee Member)
Professor Chen, Hsin Hsi (External Examiner)

Abstract

As the World Wide Web is growing exponentially, a large number of Web documents written in all kinds of languages are disseminating from many information providers. Category tree is an important way to organize these documents. The structures of category trees from different information providers are not the same but are somewhat similar. It is tedious and time consuming when users need to browse through different category tree structures from different information provides to find the information they need. So, it is desirable to integrate all source category trees into a master category tree so that users only need to browse through a unified category tree to extract information from multiple information providers. As the result of the globalization, multinational companies are running their business world wide and people can get access to all kinds of information spread in any place in the world. There are also more and more institutes and persons need to integrate category trees which are composed of documents written in different languages. Integration from multiple source category trees to a personal master category tree will support individual users to manage their documents efficiently and effectively. In this paper, we address the problem of integrating one or more source category trees to a master category tree. Our method captures the intuition that the structure of the source category tree maintains the knowledge of professional in organizing the documents. By identifying the category relationships between source category trees and master category tree and learning the inherent parent/child/sibling relationships within a category tree, we develop several decision rules to map categories in source category trees to categories in master category. In our proposed technique, we also consider integrating category trees with different languages by solving the semantic interoperability problem. As our experiments with Web data show that the proposed technique is promising in category tree integration.

内容摘要

由於互聯網的迅猛發展，許多信息提供者在網絡上不斷發布著大量用各種語言寫成的文檔。目錄樹是一個經常被他們使用來組織文檔的工具。不同的信息提供者使用的目錄樹的結構，雖然會有些相似，但往往又不盡完全相同。如果用戶需要遍歷這些不同的目錄樹，以從不同的信息提供者尋找他們需要的信息，變會覺得使用起來非常的不方便。將不同的目錄樹整合在一起的需求就此產生，它將幫助用戶更有效地利用一個統一的目錄樹來尋找不同信息提供者提供的不同信息。伴隨著全球化的進程，跨國公司們都在全球展開他們的生意，個人也可以輕易接觸到來自世界各地的信息。越來越多的機構和個人需要將存有不同語言的文檔的目錄樹整合在一起。這樣的整合，使用戶管理他們的文檔的效率大大提高。在本文中，我們主要解決了將一個或者多個源目錄樹整合到一個目標目錄樹的問題。我們的方法抓住了源目錄樹的結構還蘊含著許多專家對於文檔組織分類的專業知識這一特點，通過正確識別源目錄樹和目標目錄樹的目錄之間的關系，深度開發同一目錄樹目錄之間的父子/兄弟關系，提出了一些可以正確並有效地將源目錄樹中的目錄映射到目標目錄樹的規則。我們的技術，通過解決不同語言之間語義互釋的問題，也考慮了整合包含用不同語言寫成的文檔的目錄樹的問題。用來源於實際的網絡數據操作的實驗，也充分證明了我們方法的有效性。

Acknowledgement

I would first like to express my gratitude to my supervisor, Prof. Christopher C. Yang for his advice and guidance throughout my postgraduate study as well as this research work. Without his introducing me into the field of academic researches, I would hardly become interested in my researches.

Thanks to all the professors and students at the Department of Systems Engineering and Engineering Management, for their generous sharing of ideas, suggestions, and discussions

Thanks to The Chinese University of Hong Kong and the department of Systems Engineering and Engineering management for offering me the scholarship and the opportunity to pursue my master degrees.

I want to give my special thanks to my family members – my parents and my brother. Thank you all for supporting me in pursuing an academic path in research and understanding my ideals for life. Without you, it would be very difficult for me to persist in my way.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

The information environment of the Web is heterogeneous, decentralized and yet growing exponential. The available number of documents in digital form in the internet and intranet is so large that it is becoming increasingly difficult to find and organize relevant materials. Information explosion become a more and more serious problem to all the web users [3]. The management and retrieval of large volumes of information in the Web becomes difficult, thus lead to the information overload problem.

Many techniques are invented to help the users to search for the right information faster. Previous experiences have shown that navigate through category tree of pre-classified content is an important way for the users to search for the information they need. Document classification organizes a large collection of documents into distinct groups of similar documents – categories [21] [33]. Each category constitutes a hidden theme of the collection of documents. Document classification is defined as the task of assigning a Boolean value to each pair $< d_j, c_i > \in D \times C$, where D is a domain of documents and C is a set of predefined categories [33]. Hierarchical classification of documents is also known as category tree where documents are assigned to a category in the tree structure. They are widely used in our daily life and in many areas of computer science. Traditional library classification systems and the computing file systems adopt category trees to organize a large number of documents. In the era of internet, almost all the web directories and websites use hierarchical tree structure to organize their Web pages. The newly emerged e-business websites also adopt hierarchical structure to organize their products. Figure 1 shows how a category tree is used in Amazon. Products sold in Amazon are classified into 11 main categories and 41 smaller categories. More detailed structure of any category will be

further shown if that category is clicked by the user. For example, "Camera & Photo" can be further classified as "Digital Cameras", "Lenses", and "Film Cameras" and so on.

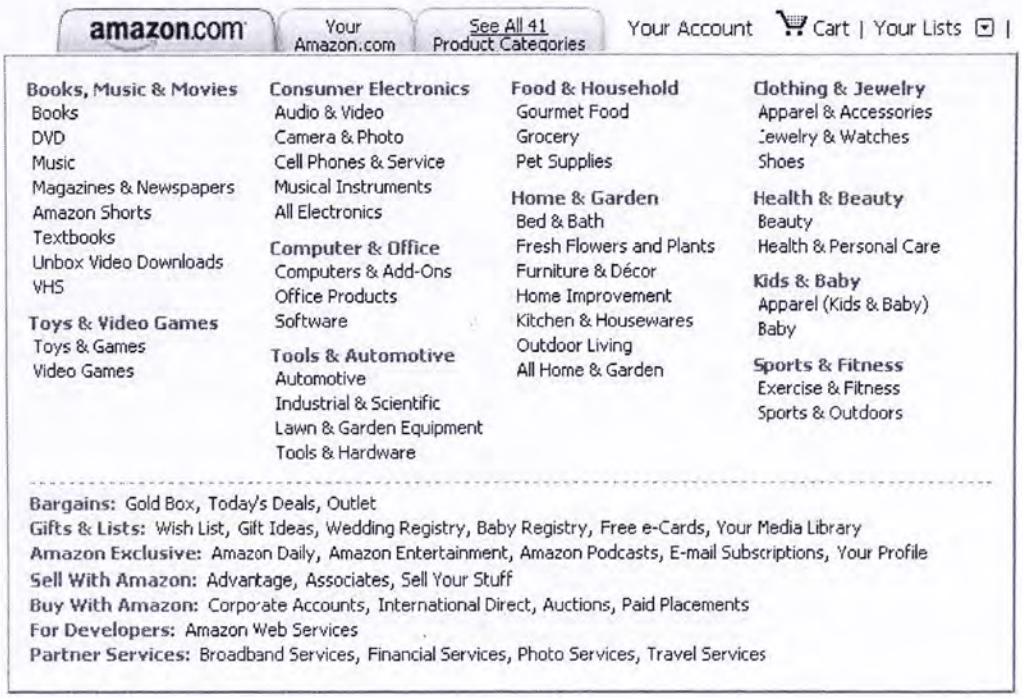


Figure 1. Example category tree used by the Amazon

Category tree is an effective and popular organization technique of documents, but different information providers or organizations usually use different hierarchical structures to organize their documents. Category tree integration arises in a variety of situations, ranging from B2B and B2C e-business, personal information management, supply chain etc. For example, Yahoo! and Google edit the directories of Web pages in their own manner; Amazon and eBay maintain two different catalogs to organize their products. Figure 2 shows how eBay organizes its products into category tree. Comparing to Figure 1, it very clear that eBay uses a structure which is totally different from that in Amazon.

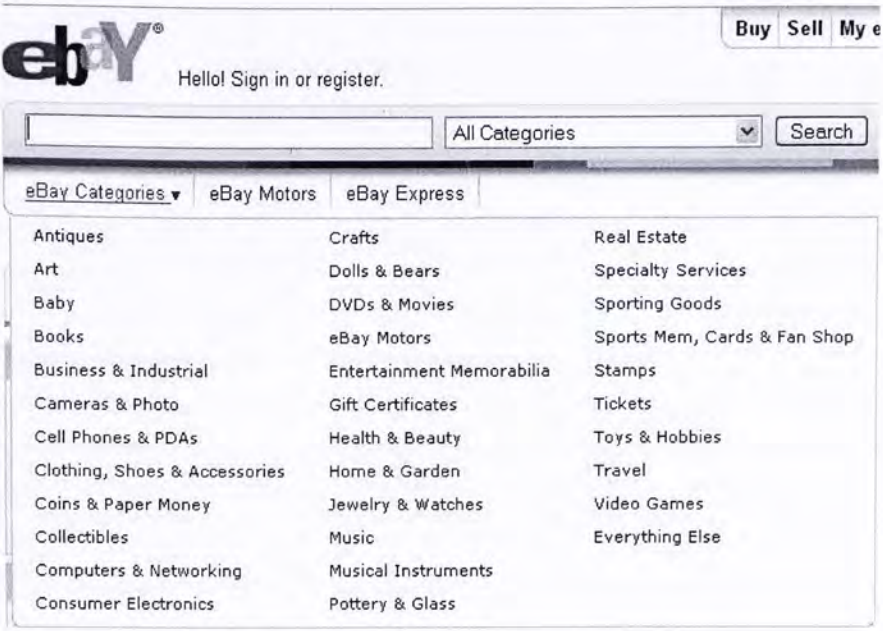


Figure 2. Example of a different structure

Given a person or an organization which has a personal category tree to organize the documents collected from multiple information sources, the documents from external information sources are likely to be organized in structures that are different from the personal category tree structure. It is necessary to integrate the source category trees to the personal category tree or a unified category tree so that users can effectively extract information from a single category tree instead of navigating through multiple category trees. For instance, a person who read news daily may want to organize news articles from CNN, BBC and other news agencies into one category tree so that he can go directly to a particular category in the master category tree and read articles in that topic from all newswires. In the business to business market, an intermediary agent may have many different suppliers and may also supply their products to different consumers. They will choose the structures which are the most suitable ones

for their specific convention and business environment instead of a unified one. So, these suppliers and consumers are very likely to use different ways to organize their products.

Except for the simple integration, in some cases, users may want to modify the master category tree based on the structures of source category trees, because the source category trees contain some knowledge that are not included in the master category tree. The master category tree may not have a structure which is as rich as the source category and therefore is not able to accommodate all the categories in the source category tree. So, it is important that the integration process is able to expand or modify the master category tree by learning the organization of documents from the source category trees. Inserting new categories, deleting old categories, splitting large categories, merging small categories or relocating categories are required into integrate the knowledge of the source category tree into the master category tree. By doing this, the category tree can evolve into a better one and become more useful and handful.

As the result of the globalization of business environments, the companies are becoming more and more multinational. The information systems of these companies have to manage knowledge obtained in multilingual from multiple geographical regions. Thus they can cooperate with companies from other countries. And as the development of internet technology, residences in one country can easily get access to websites of other courtiers. They also have to manage information and knowledge obtained from multiple sources. Institutions and individuals may need to perform category integration that involves multilingual documents. Cross-lingual category integration deals with integrating two catalogs where one contains documents written in one language (L_1) and the other consists of documents in a different language (L_2). Cross-lingual category integration evidently presents a more difficult research issue because of the language barrier between the two catalogs to be integrated.

The rest of this paper is structured as follows. A review of related work is given in Chapter 2. In Chapter 3 we formally define our problem and some terminologies. In Chapter 4, we introduce the proposed technique. Relationships between categories such as *Match*, *Disjoint*, *SubConcept*, *SuperConcept* and *Overlap* are introduced. Total 6 decision rules are developed to map categories from the source category tree to suitable positions in the master category tree. Chapter 5 presents the result of our experiments conducted using real Web data. In Chapter 6 we solve the cross-lingual category tree integration problem and show the experiment result. We conclude our work and discuss future work in Chapter 7.

Chapter 2. Related Work

Researchers in various areas of computer science have worked on data integration and sharing. No previous work is exactly the same as the problem we defined in Chapter 3. Our work is closely related to two traditional areas of research, ontology integration and schema matching. They are very similar to category tree integration, but there are also some differences between them:

- Categories in category tree integration are usually named *concepts* in ontology integration and schema matching. These concepts have many attributes and the concepts are described by assigning different values to the attributes. The value of each attribute can be complex data type. For example, a concept *Ford Explorer* in *car* ontology can have attributes: *name* (Ford Explorer), *number-of-doors* (4), *Engine* (4.0L, 4.6L), *Transmission* (6-speed). But categories in category trees are only described by their names. The only element in categories is documents. These documents are usually represented by keyword vectors.
- Concepts relationships of ontology and schema are different from category tree. The only relationship between categories is *subsumption*. Although *subsumption* is also one of the most important relationships between concepts in ontology and schema, there are many other relationships between concepts. For example, another common type of relationship is the *meronymy* relationship that represents how objects are combined together to form composite objects. And there will also be many domain-specific relations in ontology and schemas, which may not be described by trees.

From the above description, it can be seen that ontology and schema contain much richer information than category tree. But category trees usually have many instances. So, the techniques used in the ontology integration and schema matching can not be

directly applied to category tree integration. But, some of these techniques are still meaningful and useful when we are developing category tree integration techniques.

In this chapter, we will review the ontology integration and schema matching techniques first. And then, we will have a look at some category tree integration techniques which formulate this problem as text categorization. Cross-lingual category tree integration is closely related to cross-lingual text categorization (CLTC) and cross-lingual information retrieval (CLIR). We will also provide a review on existing CLTC & CLIR techniques.

2.1. Ontology Integration

Ontology is usually defined as a specification of a conceptualization. It is a data model which represents a set of concepts within a domain and the relationships between those concepts. Similar to the category trees, different parties has developed different ontologies for their own purpose. To solve the interoperability problem, it is necessary to develop ontology integration techniques. In many literatures, ontology integration is named ontology mapping, ontology merging or ontology alignment. A lot of ontology integration systems are developed by using different techniques. They can be mainly classified into 3 catalogs [9]:

- Ontology integration between an integrated global ontology and local ontologies. In this case, ontology mapping is used to map a concept found in one ontology into a view, or a query over other ontologies (e.g. over the global ontology in the local centric approach, or over the local ontologies in the global-centric approach).
- Ontology integration between local ontologies. In this case, ontology mapping is the process that transforms the source ontology entities into the target ontology entities based on semantic relation. The source and target are semantically related at a conceptual level.

- Ontology integration in ontology merge and alignment. In this case, ontology mapping establishes correspondence among source (local) ontologies to be merged or aligned, and determines the set of overlapping concepts, synonyms, or unique concepts to those sources. This mapping identifies similarities and conflicts between the various source (local) ontologies to be merged or aligned.

McGuinness et al. [27] develop the CHIMAERA, which is an interactive ontology merging and diagnosis tool based on the Ontolingua ontology development environment. They consider the task of merging two ontologies to be one of combining two or more ontologies that may use different vocabularies and may have overlapping content. It makes users affect merging process at any point during merge process, analyzes ontologies to be merged, and if linguistic matches are found, the merge is processed automatically, otherwise, further action can be made by the use. It coalesce two semantically identical terms from different ontologies so that they are referred to by the same name in the resulting ontology. It also identify terms that should be related by subsumption, disjointness, or instance relationships and provide support for introducing those tasks.

Noy et al. [29] [30] developed two semi-automated ontology merging and alignment systems named PROMPT and Anchor-PROMPT. PROMPT is developed based on the similarity of concepts which is measured by linguistic similarity. And then it generates a list of suggestions to guide the users in performing the remaining tasks based on linguistic and the internal structure of the concepts and their position in the ontology. Anchor-PROMPT provide additional possible points of similarity between ontologies. It takes as input a set of related terms – anchors – from their source ontologies. It then traversed the paths between the anchors in the corresponding ontologies. A path follows the links between classes defined by the hierarchical relations or by slot and their domains and ranges. It then compares the terms along these paths to produces a set of new pairs of semantically close terms.

Ryutaro et al. [32] proposed a system named HICAL (Hierarchical Concept Alignment system). HICAL provides concept hierarchy management for ontology merging. It proposes a new method that allows a concept in one concept hierarchy to be aligned with another concept in another concept hierarchy. It uses machine learning method (k statistic) to measure the similarity between concepts. The algorithm chooses the most similar one for integration but the relationships between the concepts and the hierarchical structure are not used. It exploits the data instances in the overlap between the two taxonomies to infer mappings. It uses hierarchies for categorization and syntactical information, not similarity between words, so that it is capable of categorizing different words under the same concept.

Doan et al. [14] attempted to match ontologies on the semantic Web. Their approach is embodied in a system named GLUE, which is a system that employs machine learning techniques to find such mappings. Given two ontologies, for each concept in the ontology, GLUE finds the most similar concept in the other ontology. It uses multiple learning strategies, each of which exploits well a different type of information either in the data instances or in the taxonomic structure of the ontologies. Glue has a total of three learners: Content Learner, Name Learner, and Meta Learner. Content and Name Learners are two base learners, while Meta Learner combines the two base learners' prediction. Glue also tries to incorporate commonsense knowledge, a variety of heuristic knowledge, and domain-specific constraints by relaxation labeling.

Su and Gulla [36] present a heuristic mapping method and a prototype mapping system that support the process of semi-automatic ontology mapping for the purpose of improving semantic interoperability in heterogeneous systems. It is an information retrieval approach based on the idea of semantic enrichment, i.e., using instance information of the ontology to enrich the original ontology and calculate similarities between concepts in two ontologies. Some adjustments are made based on the name of the categories. They use WordNet to judge the similarity of names.

2.2. Schema Matching

Schema matching is a basic problem in many database application domains. It takes two schemas as input and produces a mapping between elements of the two schemas that correspond semantically to each other. In general it is not possible to determine fully automatically matches since most schemas have semantics that affect the matching criteria but is not formally expressed or documented. Usually, the following largely-orthogonal classification criteria of schema matching are considered [31]:

- Instance vs. Schema: Schema-level matchers only consider schema information, not instance data. The available information includes the usual properties of schema elements, such as name, description, data type, relationship types (part-of, is-a, etc.), constraints, and schema structure. Instance-level data can give important insight into the contents and meaning of schema elements. It is very useful when useful schema information is limited.
- Element vs. Structure Matching: Element-level matching techniques compute mapping elements by analyzing entities in isolation, ignoring their relations with other entities. Structure-level techniques compute mapping elements by analyzing how entities appear together in a structure.
- Language vs. Constraint: a matcher can use a linguistic based approach (e.g., based on names and textual descriptions of schema elements) or a constraint-based approach (e.g., based on keys and relationships).
- Matching Cardinality: the overall match result may relate one or more elements of one schema to one or more elements of the other, yielding four cases: 1:1, 1:n, n:1, n:m. In addition, each mapping element may interrelate one or more elements of the two schemas. Furthermore, there may be different match cardinalities at the instance level.
- Auxiliary Information: most matchers rely not only on the input schemas S_1 and S_2 but also on auxiliary information, such as dictionaries, global schemas, previous matching decisions, and user input.

Berlin and Motro [6] described an approach that makes use of Bayesian learning approach to acquire probabilistic knowledge from examples of schemas that have been "mapped" by domain experts into a knowledge base of database attributes call attribute dictionary. The "match score" between all the attributes of two schemas is measured based on the attribute dictionary. Then optimal matching between these schemas is selected out of all the possible combinations.

Cupid proposed by J. Madhavan et al. [26] implements a hybrid matching algorithm comprising linguistic and structural schema matching techniques, and computes similarity coefficients with the assistance of a domain specific thesaurus. Input schemas are encoded as graphs. Nodes represent schema elements and are traversed in a combined bottom-up and top-down manner. The matching algorithm consists of three phases and operates only with tree-structures to which non-tree cases are reduced. The first phase (linguistic matching) computes linguistic similarity coefficients between schema element names (labels) based on morphological normalization, categorization, string-based techniques (common prefix, suffix tests) and a thesaurus look up. The second phase (structural matching) computes structural similarity coefficients weighted by leaves which measure the similarity between contexts in which elementary schema elements occur. The third phase (mapping elements generation) computes weighted similarity coefficients and generates final alignment by choosing pairs of schema elements with weighted similarity coefficients which are higher than a threshold.

F. Giunchiglia et al. [15] proposed a system named S-Match, which is a schema-based matching system. It takes two graph-like structures (e.g., XML schemas) and returns semantic relations (e.g., equivalence, subsumption) between the nodes of the graphs that correspond semantically to each other. The relations are determined by analyzing the meaning (concepts, not labels) which is codified in the elements and the structures of schemas. In particular, labels at nodes, written in natural language, are translated into propositional formulas which explicitly codify the label's intended

meaning. This allows for a translation of the matching problem into a propositional unsatisfiability problem, which can then be efficiently resolved using (sound and complete) state of the art propositional satisfiability deciders. S-Match was designed and developed as a platform for semantic matching, namely a highly modular system with the core of computing semantic relations where single components can be plugged, unplugged or suitably customized. It is a hybrid system with a composition at the element level. At present, S-Match libraries contain 13 element-level matchers, and 3 structure-level matchers.

Techniques used in the ontology integration and schema matching are very similar. From our previous review, it is clear that all of these techniques suffer several serious shortcomings:

- They didn't make full use of the hierarchical structure information such as parent/child/sibling relationships contained in the trees. Hierarchical structure is only used to increase the accuracy of the measurement of concept similarity in previous work [26] [30]. Even more, some ontologies or schemas are transformed into flat ones for integration. Sometimes some unreasonable results may appear, for example, the parent concept A of concept B in the source ontology/schema may become the child concept of B in the master ontology/schema.
- Previous work tried to integrate the ontology/schema of based on the symmetric concept similarity only, but the real case is that the relationships between concepts are asymmetric. They didn't further identify and classify these relationships among concept to make the integration more accurate.
- Learning from the source ontology/schema cannot be achieved to expand the master ontology/schema. They focus on how to integrate the source ontologies or schemas only. Master ontologies or schemas are static in their algorithms and can not be improved. Expert knowledge contained in the source ontology/schema is discarded after the integration.

2.3. Taxonomy Integration as Text Categorization

Although most of the work has been focused on ontology mapping or schema mapping, there is some recent work on category tree mapping. They formulate it as text categorization problem. These methods make use of the documents categorization information in the source category to improve the classical text classification model.

Agrawal and Srikant [2] first introduced the problem of integrating documents from different resources into a master catalog. They squeezed the hierarchical structures of the catalogs into flat structure and extend Naïve Bayes approach to build more accurate classification models by using the implicit document similarity in the source catalog. The basic naïve Bayes algorithm is enhanced. It use $\Pr(C_i \mid d, S)$ instead of $\Pr(C_i \mid S)$ to estimated which category the document d should belong to, and $\Pr(C_i \mid d, S) = \Pr(C_i \mid S)\Pr(d \mid C_i)/\Pr(d \mid S)$. It will first classify the documents using the basic algorithm to estimate $\Pr(C_i \mid S)$.

D. Zhang and W.S. Lee [46] enhanced the Support Vector Machine (SVM) algorithm to attack the problem. Their key insight is that it would be beneficial to do transductive learning rather than inductive learning, i.e., learning to optimize classification performance on a particular set of test examples. Noticing that the categorization of the master and source taxonomies often have some semantic overlap, they propose a method, Cluster Shrinkage (CS), to further enhance the classification by exploiting such implicit knowledge. Their integration approach first applied CS on all objects in M and N, then trained TSVMs on these objects, finally used the learned TSVMs to classify the objects in N into the categories in M. They name this approach CS-TSVM.

Wei and Cheng [8] [37] proposed a clustering-based approach for category integration to handle heterogeneities among coarse grained categorization. Each

source category is divided into several subcategories. These subcategories are then merged into most similar master categories.

These techniques try to improve the traditionally text classification methods to solve the integration problem. They integrated the source category tree document by document. The hierarchical structures of the source category trees are totally discarded and the structure of the master category tree can not be improved again. Figure 3 illustrates how the hierarchical structures of two trees are removed and two sets of categories are formed. There are no relationships between the categories in the sets and all the categories in the sets are treated by the integration algorithm equally. It is possible that the ancestor-descendant relationships between categories are destroyed after the integration. For example, documents in the category *B* of category set *S* may be mappd to the category *e* of category set *M* and documents in category *D* of the category set *S* may be mapped to the category *b* in the category set *M*. Category *B* is the parent of category *D* in the category tree *S* before the integration, but it becomes the child of category *D* after the integration.

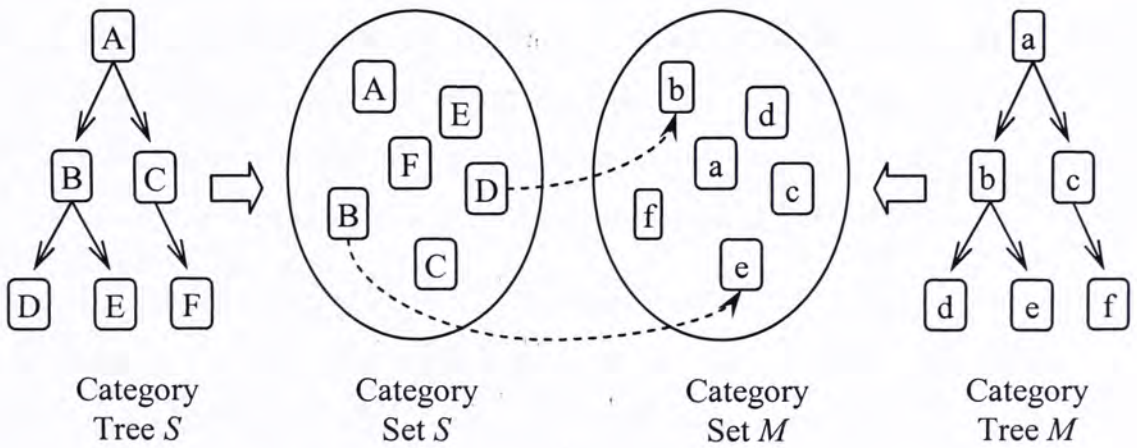


Figure 3. Removing the structure of category trees

2.4. Cross-lingual Text Categorization & Cross-lingual Information Retrieval

Cross-lingual information retrieval (CLIR) allows users to retrieve documents in a language different from a query language made in the users' own language. There are four commonly employed approaches for Cross-lingual information retrieval [1] [39]:

- Machine Translation Approach: This approach uses machine translation system to translate foreign language documents in the corpora or the users query into the same language.
- Dictionary-based query translation Approach: In dictionary based query translation the query keywords are translated to the target language using machine readable dictionaries.
- Translation Disambiguation: Translation ambiguity arises due to source and target language lexical ambiguity. Many methods have been developed to decrease the ambiguity in query translation, such as part of speech tagging, corpus based disambiguation methods, query structuring, et al.
- Corpus-based Approach: A corpus is a repository of a collection of natural language material. It makes use of the statistical information of term usage in a parallel or comparable corpus to automatically construct a statistically based cross-lingual thesaurus to overcome the limitations of other approaches.

Cross-lingual text categorization (CLTC) learns from a set of training documents in one language and classifies new documents in other languages [7]. It is a new research subject which employed almost the same techniques as CLIR. Three translation strategies may be distinguished:

- Document Translation: Although translating the complete document is workable, it is not popular in CLIR, because automatic translations are not satisfactory and manual translations are too expensive.

- Terminology Translation: constructing a terminology for each of the relevant domains (classes), and translating all domain terms. It is expected that these include all or most of the terms which are relevant for classification.
- Profile-based Translation: translate only the most important terms actually occurring in the class profiles.

Our work about the category tree integration attempts to overcome the shortcoming of the current work. The contributions can be summarized as:

- Extend the Bayes rules to determine the category relationships between categories from different category trees.
- Develop six decision rules to map a category from the source category tree to a category in the master category tree.
- Develop integration technique that satisfies the constraints imposed by the original structures of the source category trees and the structure of the personal master category tree.
- The integration technique is able to expand or modify the master category tree by learning the organization of documents from the source category trees.
- The integration techniques in not limited to mono-lingual category tree integration. Our experiment has shown that it can also be applied to cross-lingual category tree integration after solving the semantic interoperability problem.

Chapter 3. Problem Definition

3.1. Mono-lingual Category Tree Integration

We now formally define the category tree integration problem we are solving. The research problem is defined as integrating one or more source category trees to a master category tree. A category tree, T, has a set of categories with certain hierarchical structure. It can be represented as $T = \{C, E\}$.

$C = \{C_1, C_2, ..., C_{|C|}\}$ is defined as the set of categories in T, and a set of documents $D_i = \{d_{1i}, d_{2i}, ..., d_{|D_i|i}\}$ are assigned to each category, $D_i \in C_i$. D_i can be an empty set. The names of the categories are temporarily omitted in this study. Documents in the set D_i contain the textual content and are assigned to one or more categories. In general, each document is represented as term vector $\vec{d}_i = < w_{1i}, w_{2i}, w_{3i}, ..., w_{ti} >$, where each dimension represent the weight of a term obtained from preprocessing by vector space model. C_{root} is defined as the *root category* of the category tree, $C_{root} \in C$. Categories which have no child nodes are called *leaf categories*. All categories except for leaf category and C_{root} are called *inner categories*.

$E = \{E_1, E_2, ..., E_{|E|}\}$ is the set of ordered pairs called edges (C_p, C_q). $(C_p, C_q) = E_i \in E \in \{C \times C\}$. The direction is defined from the parent node C_p to the child node C_q, specified through the relational operator $C_p \rightarrow C_q$ which is also called direct path from C_p to C_q. A path $C_i \rightarrow C_j$ with length $j - i$ is therefore an ordered set of nodes $\{C_i \rightarrow C_{i+1} \rightarrow, ..., C_{j-1} \rightarrow C_j\} \subset C$, where each node is the parent nodes of the following node. In category tree with a path $C_i \rightarrow C_j$, there exists no path $C_j \rightarrow C_i$ since the tree is acyclic. To describe the relationship of category C_i

with other categories in category tree T, some special notations are defined as follows:

- $Parent(C_i)$: the set of parent category of C_i, $Parent(C_i) \subset C$;
- $Children(C_i)$: the set children of categories of C_i, $Children(C_i) \subset C$;
- $Descendent(C_i)$: the set of descendent categories of C_i, $Descendent(C_i) \subset C$;
- $Ancestor(C_i)$: the ancestor categories of C_i, $Ancestor(C_i) \subset C$.

There exist several characteristics of the category tree:

- There is one and only one root category, C_{root}, which is the super concept of all the categories in the category tree
- For $\forall C_i$, $C_i \neq C_{root}$, $|Parent(C_i)| = 1$
- For $\forall C_i$, $|Children(C_i)| \in [0, |T|-1]$, $|Descendent(C_i)| \in [0, |T|-1]$
- $Parent(C_i)$ subsumes C_i, C_i subsumes all he categories in $Children(C_i)$, this is the only relationships between categories.

For category tree integration problem, there exists a source category tree $T^s = \{C^s, E^s\}$ and a master category tree $T^m = \{C^m, E^m\}$. For any category C_i in C^s, with $D_i = \{d_{1i}, d_{2i}, ..., d_{|D_i|i}\} \in C_i$, it may be added as a new category in C^m, or be merged with C_j in C^m to form a new category in C^m, or be split as several categories and then added or merged. Any way, all the documents in the source category tree T^s will be assigned to one of the categories in T^m. For edges in E^s, it is much more simple. E^s is only used to help to make E^m more comprehensive, meaningful and useful. Most of them will be discarded and some of them will be integrated into E^m. Besides, new edges may also be created and added to E^m.

The objective of the integration is to put the categories in the source category tree to the proper positions in the master category tree. But, we will also try to maintain the structures of the source category trees in the master category tree after integration.

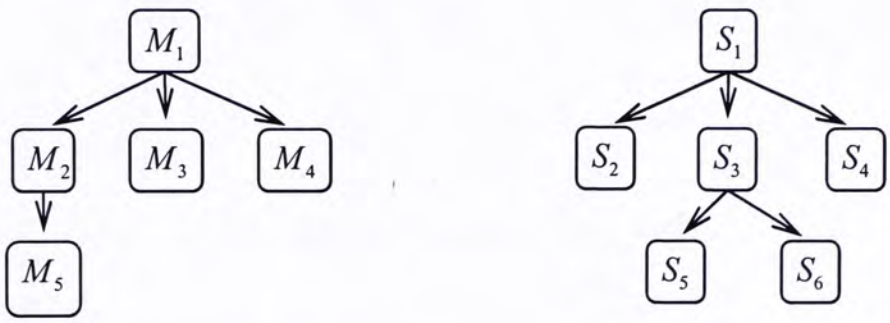
Documents in the source category trees are organized based on the knowledge of experts' of the information providers. Such organization is usually more accurate than machine generated category tree based on feature extraction and machine learning because the professionals who generate the source category trees have the domain knowledge of the source documents. Maintaining the structure of the source category trees will retain the knowledge of the information provider in the master category tree. During the integration, a little adjustment may be made to the master category tree if necessary, but we will try to make as little adjustment as possible.

3.2. Integration Operators

In order to finished the complicated integration tasks described in the previous paragraphs, 3 integration operators are defined as follows. These integration operators focus on the categories of the source category tree, C_i^s. The three operators also form the foundation of one of our evaluation method.

- *Map*: C_i^s may be mapped to an existing category C_j^m in the master category tree, then all the documents labeled C_i^s should also be labeled C_j^m, noted as $Map(C_i^s; C_j^m)$; or
- *Insert*: C_i^s may be inserted as a newly expanded category in the master category tree, C_{new}^m, then all the document labeled C_i^s should be also labeled C_{new}^m, noted as $Insert(C_i^s; C_{new}^m, C_{parent}^m, C_{child}^m)$, where C_{parent}^m is the parent of C_{new}^m and C_{child}^m is the child of C_{new}^m; if C_{child}^m is omitted, C_{new}^m is inserted as a leaf category ; or
- *Split*: C_i^s may be spitted into several sub-categories, $C_{i_1}^s, C_{i_2}^s, \ldots, C_{i_n}^s$, noted as $Split(C_i^s; C_{i_1}^s, C_{i_2}^s, ..., C_{i_n}^s)$. After that, each of the sub-categories will be mapped to an existing category or inserted as an expanded category.

Map, Insert and Split are three integration operators, we will explain when and how to use these operators in detail in Chapter 4.



Master Category Tree　　Source Category Tree

Figure 4. The category trees that will be integrated

Let us give a simple example to further explain our definition. Figure 4 shows a source category tree and a master category tree that will be integrated together. $C^m = \{M_1, M_2, M_3, M_4, M_5\}$, $E^m = \{(M_1, M_2), (M_1, M_3), (M_1, M_4), (M_2, M_5)\}$. $C^s = \{S_1, S_2, S_3, S_4, S_5, S_6\}$, $E^s = \{(S_1, S_2), (S_1, S_3), (S_1, S_4), (S_3, S_5), (S_3, S_6)\}$. In this example, we will first *Map* S_1 to M_1, S_2 to M_2. S_3 to M_3. And then, we will *Split* S_4 into two categories, S_{4_1} and S_{4_2}. After that, we will *Map* S_{4_1} to M_4 and *Insert* S_{4_2} as new category. At last, we will *Intert* S_5 and S_6 as new categories.

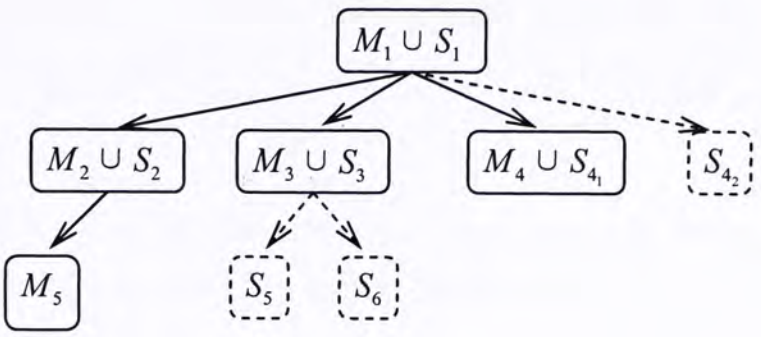


Figure 5. Integration result

Figure 5 shows the integration result. $T^{new} = \{C^{new}, E^{new}\}$. C^{new} and E^{new} have 8 and 7 elements in them respectively.

3.3. Cross-lingual Category Tree Integration

The documents in the categories can be written in different languages, let $L = \{L_1, L_2, ..., L_{|L|}\}$ be a set of languages. If the documents in the two category trees, $T^s = \{C^s, E^s\}$ and $T^m = \{C^s, E^s\}$, use more than one languages, then the category tree integration problem changed from mono-lingual to cross-lingual. For the reason of simplicity, we reduce the multi-lingual case with k languages to *k-1* bi-lingual problems, thus studying the bi-lingual case is not restrictive with respect to the multi-lingual problem. We denote the two languages with L_1 and L_2. Three types of practical cases in the cross-lingual category tree integration can be distinguished.

- All the documents within one category tree are written in the same language, but the source and master category trees use different languages. In this case, category trees can be denoted as $T^{L_i} = \{C^{L_i}, E^{L_i}\}$, $i \in \{1,2\}$
- All the documents with one category are written in the same language, but different categories in the same category tree use different languages. In this case, the set of categories can be denoted as $C = \{C_1^{L_i}, C_2^{L_i}, ..., C_{|C|}^{L_i}\}$, $i \in \{1,2\}$

- The entire document uses the same language, but different documents in the same category are written in different languages. In this case, the set of documents can be denoted as $D_i = \{d_{1i}^{L_i}, d_{2i}^{L_i}, ..., d_{|D_i|i}^{L_i}\}$, $i \in \{1, 2\}$

The heart of multilingual knowledge management, including cross-lingual information retrieval, cross-lingual text classification, is the cross-lingual semantic interoperability. Among the three cross-lingual category tree integration types, the most important one is the third one. If any documents represented in one language can be interpreted by the other language, the other two types can be easily solved by using the same technique.

Chapter 4. Mono-lingual Category Tree Integration Techniques

In this chapter we present our proposed techniques for the mono-lingual category tree integration. Firstly, our method is based on correctly identifying relationship among categories. After that, six decision rules are developed to exploit hierarchical tree structure. At last, a top-down level-based algorithm makes use of the relationships and rules to integrate the category trees.

4.1. Category Relationships

Correctly identifying category relationship is the foundation of the integration algorithm. Because the relationships of the categories among the same category tree are very clear from the existing tree structure, we will focus on the relationships of the categories between different category trees. For example, the relationship between A in T^s and B in T^m. S. Zhu et al. [47] have some initial discussion about this problem. The mapping algorithm is based on the relationships between categories in the master and source category tree, so we have to clearly identify them first. Categories can be regarded as a set of documents, so set theory can be lent to define our category relationships. If we choose one category node from source and master category respectively, e.g. A in T^s and B in T^m, the relationships between the two categories can be classified as follows. Figure 6 shows how to represent the category relationships in the Venn diagram.

- *Match*: $\forall d,\ d \in A \leftrightarrow d \in B$, which means all the documents labeled A are also labeled B, and vice versa. We call it *A match B* and denote it as $Match(A, B)$.

- *Disjoint :* $\forall d,\ (d \in A) \wedge (d \in B) = FALSE$, which means only part of the documents labeled A are also labeled B, and vice versa. We call it *A disjoint B* denote it as $Disjoint(A, B)$.
- *SubConcept / SuperConcept* , here we assume category A is subsumed by category B. Then, $\forall d,\ d \in A \rightarrow d \in B$, which means all the documents labeled A are also labeled B, but only part of documents labeled B are also labeled A. We call it *A SubConcept B* or *B SuperConcept A* and denote it as $SubConcept(A, B)$ or $SuperConcept(B, A)$.
- *Overlap* , $\exists d,\ (d \in A) \wedge (d \in B) = TRUE$, which means some of the documents labeled A are also labeled B and some of the document labeled B are also labeled A. We call it *A overlap B* or *B overlap A* and denote it as $Overlap(A, B)$. Please note that *Overlap* relationship doesn't include those special cases, such as *Match* and *SubConcept/SuperConcept.*

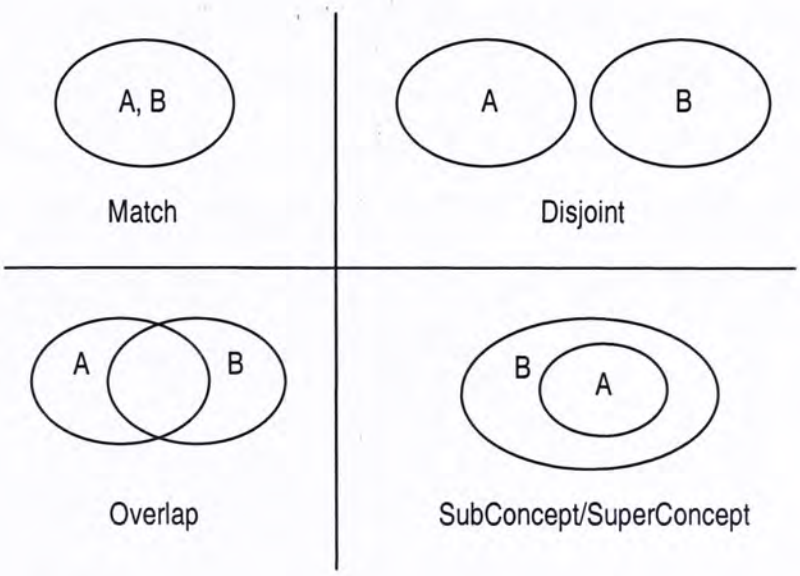


Figure 6. Venn diagram of category relationships

Among these five relationships, *Match*, *Disjoint* and *Overlap* are symmetric, which means $Match(A, B) = Match(B, A)$, $Disjoint(A, B) = Disjoint(B, A)$ and

$Overlap(A,B) = Overlap(B,A)$; For *SubConcept* and *SuperConcept*, $SubConcept(A,B) = SuperConcept(B,A)$.

Category relationships are determined by the documents that belong to them, for the documents are the basic elements of categories. We adopt the conditional probability formula $P(A \mid B)$, to determine the category relationships:

$$P(A \mid B) = \frac{number\ of\ documents\ labeled\ B\ predicted\ to\ be\ labeled\ A}{number\ of\ documents\ labeled\ B}$$

We can simply count the "*number of documents labeled B*" in category *B*. "*number of documents labeled B predicted to be labeled A*" can be determined by automated text classifier. Please note that in the path $C_p \rightarrow C_q$ of the tree structure, all the documents labeled C_q are also regarded as labeled C_p.

Based on our definition, if $Match(A,B)$, then $P(A \mid B) = 1$ and $P(B \mid A) = 1$. But automated text classifiers will always produce some errors and the ideal case will never be reached. So, if we want to use $P(A \mid B)$ and $P(B \mid A)$ to determine the category relationships, two parameters th_H and th_L need to be added. th_H should be a little smaller than 1 and th_L should be a little larger than 0. Then, the relationships between categories can be determined as follows:

- $Match(A,B)$:

  $P(A \mid B) \geq th_H \wedge P(B \mid A) \geq th_H$

- $Disjoint(A,B)$:

  $P(A \mid B) \leq th_L \wedge P(B \mid A) \leq th_L$

- $SubConcept(A,B)$:

  $P(A \mid B) < th_H \wedge P(B \mid A) > th_H$

- *SuperConcept(A,B)* :

$$P(A \mid B) > th_H \wedge P(B \mid A) < th_H$$

- *Overlap(A,B)* :

$$th_L < P(A \mid B) < th_H \wedge 0 < P(B \mid A) < th_H$$

$$or\ 0 < P(A \mid B) < th_H \wedge th_L < P(B \mid A) < th_H$$

Figure 7 shows how to determine the category relationships in a two dimensional coordinate diagram by representing $P(B \mid A)$ and $P(A \mid B)$ in *x-axis* and *y-axis* respectively.

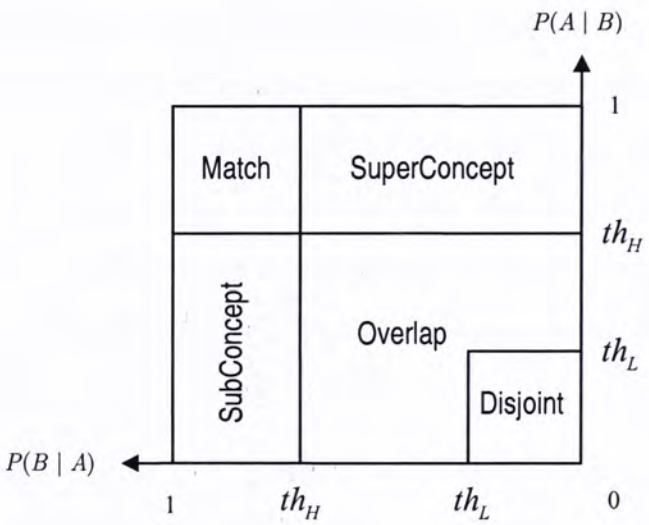


Figure 7. The category relationships in the coordinate diagram

Category relationships are the foundation of our integration technique. The decision rules defined in the next section are built based on the category relationships. Different category relationships combined with different structures form various scenario. We use decision rules to formulate these scenarios. The rules make use of the category relationships to tell us all the possible ways of what positions categories should be mapped to, what positions categories should be inserted and how a category should be splite. Category relationships are also very useful in the integration

algorithm. They narrow the search scope of the algorithm and decide the priority order of the rules to be fired if the firing conditions of several rules are satisfied at the same time.

4.2. Decision Rules

Rule learning methods usually attempt to select the best from all possible covering rules according to some minimality criterion in decision rule classifiers [33]. Several rule learners have been developed for document classifications [10] [23]. In decision rules classifiers, a set of logical rules are defined (one per category) in the form of *if <disjunctive normal form formula> then <category>* [33]. In this work, we develop rules to integrate categories from source category trees to a master category tree. These rules tell us what position a category in the source category should be placed in the master category tree. The objectives of the rules are to maintain the structure of the source category tress while integrating with the master category tree. The conditions to fire the rules are the mapping between categories in the source category tree and the master category tree and the relationships among the categories in the source category tree.

We have developed six rules in total. Rule one to Rule four are the core of them, because they decide directly what position a category of the source category tree should be placed in the master category tree. A very important assumption contained in Rule one to Rule four is that the ancestor-descendant relationship should be maintained. If category A is the descendant of category B in the source category tree, this relationship should be maintained in the master category tree after the integration. A will never turn out to be the ancestor or sibling of B, and A will not become the descendant of the siblings of B either. This assumption is a very common and easy to understand, but it is very important. It also forms the theory foundation of *Candidate Categories* of the integration algorithm which will be discussed in Section 4.3. Rule five decides when and how a category should be split, and then Rule one to Rule four

will be applied to the split categories. Rule six will make some adjustments of Rule one to Rule four.

1) Rule One: Merging to the Corresponding Category

Definition (Figure 8): Given $Map(S_j, M_i)$, $S_j^{'} \in Children(S_j)$ and $M_i^{'} \in Descedant(M_i)$, if $Match(S_j^{'}, M_i^{'})$, then $Map(S_j^{'}, M_i^{'})$.

Justification: if the category $S_j^{'}$ in the source category tree can find a category $M_i^{'}$ in the master category tree which *match* it within the *Candidate Categories*, then all the documents labeled $S_j^{'}$ are also labeled $M_i^{'}$.

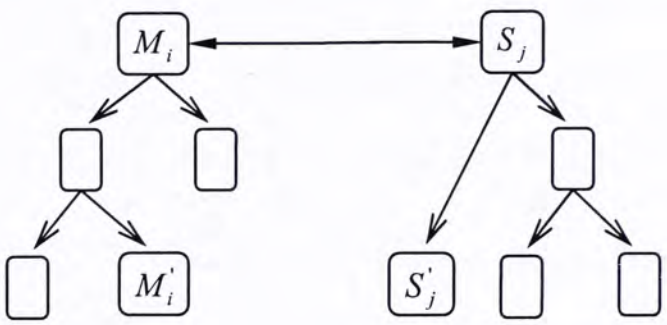


Figure 8. Rule one: Merging to the corresponding category

2) Rule Two: Expanding With a New Branch

Definition (Figure 9): Given $Map(S_j, M_i)$, $S_j^{'} \in Children(S_j)$, if $\forall M_j \in Decencent(M_i)$, $Disjoint(S_j^{'}, M_j)$, then $Insert(S_j^{'}, M_{new}, M_i)$, and all the descendants of $S_j^{'}$ are also inserted as the descendants of $S_j^{'}$.

Justification: if the category $S_j^{'}$ in the source category tree *disjoint* with all the categories within the *Candidate Categories* in the master category tree, then a new category M_{new} should be created and placed in a suitable position.

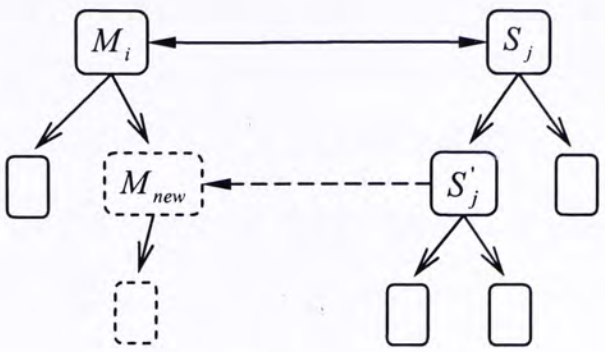


Figure 9. Rule two: Expanding with a new branch

3) Rule Three: Expanding As a SubConcept

Definition (Figure 10): Given $Map(S_j, M_i)$, $S_j' \in Children(S_j)$ and $M_i' \in Descedant(M_i)$, if $SubConcept(S_j', M_i')$, then $Insert(S_j', M_{new}, M_i')$, and all the descendants of S_j' is also inserted as the descendants of S_j'.

Justification: if the category S_j' in the source category tree can find a category M_i' in the master category tree which *SupConcept* it within the *Candidate Categories*, then a new category M_{new} should be created and placed in a suitable position.

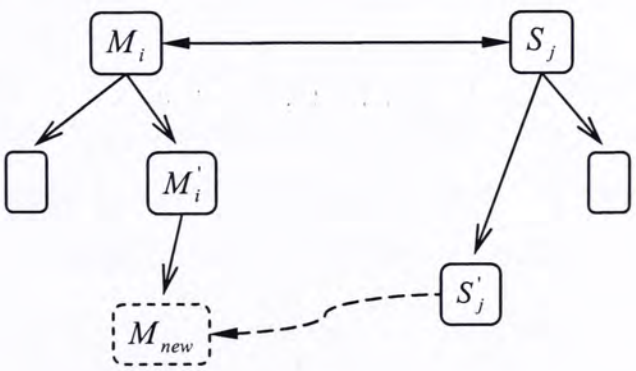


Figure 10. Rule three: Expanding as a subconcept

4) Rule Four: Expanding As A SuperConcept

Definition (Figure 11): Given $Map(S_j, M_i)$, $S_j^{'} \in Children(S_j)$ and $M_i^{'} \in Descedant(M_i)$, if $SuperConcept(S_j^{'}, M_i^{'})$, then $Insert(S_j^{'}, M_{new}, M_i, M_i^{'})$, and all the descendants of $S_j^{'}$ is also inserted as the descendants of $S_j^{'}$. The edge $(M_i, M_i^{'})$ should be deleted.

Justification: if the category $S_j^{'}$ in the source category tree can find a category $M_i^{'}$ in the master category tree which *SubConcept* it within the *Candidate Categories*, then a new category M_{new} should be created and placed in a suitable position.

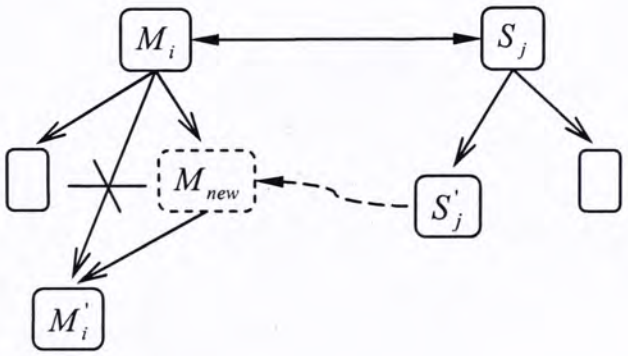


Figure 11. Rule four: Expanding as a superconcept

Figure 12 and Figure 13 give out two category trees from the real case, we will use them to illustrate how Rule one, Rule two, Rule three and Rule four are applied. In this example, the master category tree (Figure 12) is part of Yahoo! directory and the source category tree (Figure 13) is part of the Open Directory Project directory.

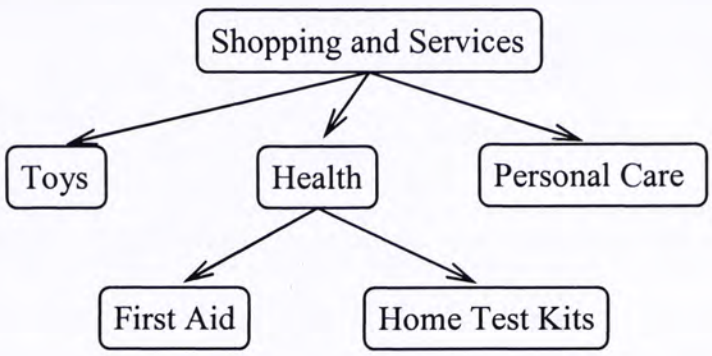


Figure 12. Master category tree of the example

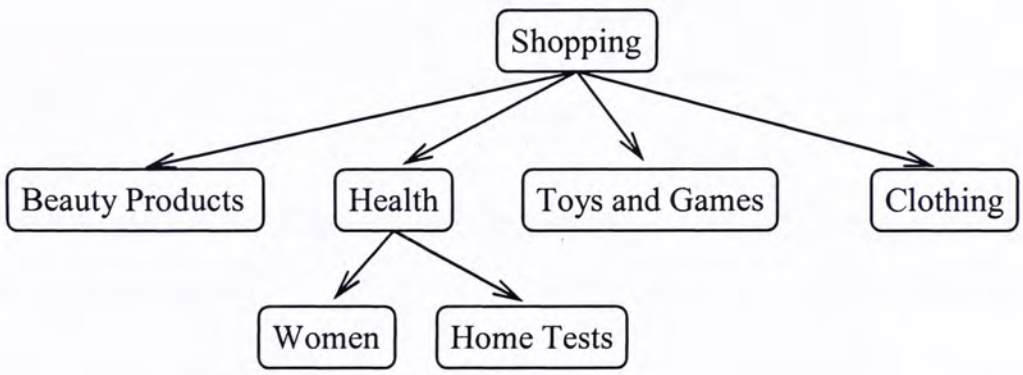


Figure 13. Source category tree of the example

We assume that the category relationships between all the categories in the source and master category tree are already known by using text classifiers before applying these rules. The category relationships between the categories are listed in Table 1. The second column of the table is the categories of the master category tree and the second row of the table are the categories of the source category tree.

Table 1. Category relationships between the categories

		Master Category Tree					
		Shopping and Services	**Toys**	**Health**	**Personal Care**	**First Aid**	**Home Test Kits**
Source Category Tree	**Shopping**	Match	SuperConcept	SuperConcept	SuperConcept	SuperConcept	SuperConcept
	Beauty Products	SubConcept	Disjoint	Disjoint	SubConcept	Disjoint	Disjoint
	Health	SubConcept	Disjoint	Match	Disjoint	SuperConcept	SuperConcept
	Toys and Games	SubConcept	SuperConcept	Disjoint	Disjoint	Disjoint	Disjoint
	Clothing	SubConcept	Disjoint	Disjoint	Disjoint	Disjoint	Disjoint
	Women	SubConcept	Disjoint	Disjoint	Disjoint	Disjoint	Disjoint
	Home Tests	SubConcept	Disjoint	SubConcept	Disjoint	Disjoint	Match

Here is how the source category tree is integrated into master category tree by using the rules we developed:

- "Shopping" *match* "Shopping and Services", so "Shopping" is mapped to "Shopping and Services".
- "Clothing" is the child of "Shopping" and it *disjoints* with all the descendant categories of "Shopping and Services", so according to Rule two, "Clothing" will be inserted as a new child of "Shopping and Service".
- "Beauty Products" and "Personal Care" are the child category of "Shopping" and "Shopping and Services" respectively, and "Beauty Products" *subconcept* "Personal Care", so it will be inserted as a new child of "Personal Care" based on Rule three.
- "Toys and Games" and "Toys" are the child category of "Shopping" and "Shopping and Services" respectively, and "Toys and Games" *superconcept* "Toys", so it will be inserted as a new child of the "Shopping and Servieces" and the parent of "Toys" according to Rule four.

- Both "Shopping" and "Shopping and Services" have a child named "Health", and they *match*. So, based on Rule one, "Health" of the source category tree will be mapped to "Health" of the master category tree.
- "Women" is the child of "Health" in the source category tree and *disjoint* with all the descendants of "Health" in the master category tree so it will be inserted as a new child of "Health" in the master category tree according to Rule two.
- "Home Tests" and "Home Test Kits" are the child of "Health" in the source and master category trees respectively and they *match*. So, based on Rule one, "Home Tests" will be mapped to "Home Test Kits"

The integration result is shown in Figure 14. All the categories in the source category tree are placed to the appropriate positions in the master category tree, and the structure of the source category is also slightly adjusted.

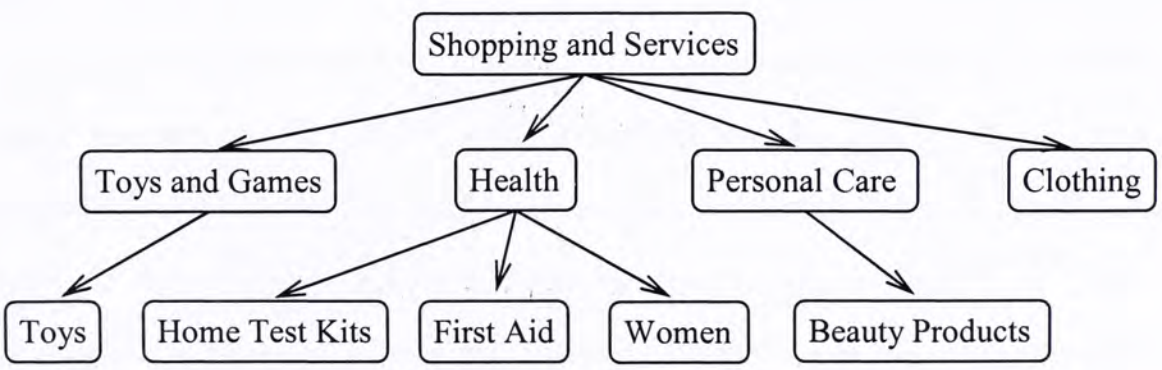


Figure 14. Integration result

Rule one to Rule four is the basic and the core rules to do category integration, but they do not cover the "Overlap" relationship between categories. So, we need another rule to split these categories.

5) Rule Five: Splitting the Category

Definition: Given $Map(S_j, M_i)$, $S_j^{'} \in Children(S_j)$ and $M_i^{'} \in Descedant(M_i)$, if $Overlap(M_i^{'}, S_j^{'})$ and $\nexists M_i^{*}$, which make $Match(M_i^{*}, S_j^{'})$ or $SubConcopt(M_i^{*}, S_j^{'})$ or $SubConcept(M_i^{*}, S_j^{'})$, then $Split(S_j^{'}; S_{j_1}^{'}, S_{j_2}^{'})$, $S_{j_1}^{'}$ and $S_{j_2}^{'}$ are further processed by Rule one to Rule four.

Justification: If a category $S_j^{'}$ in the source category tree disjoint with one or more categories in the master category within the *Candidate Categories*, and $S_j^{'}$ doesn't match, subconcept or superconcept with any categories, then $S_j^{'}$ should be split into several categories, and then these newly created categories will be placed to suitable positions based on Rule one to Rule four.

We have to use automatic text classifier again in order to split $S_j^{'}$. The number of categories that *overlap* with $S_j^{'}$ may be one or several. Let $M_k = \{M_{1k}, M_{2k}, ..., M_{nk}\}$ be the set of categories that overlap with $S_j^{'}$. A binary classifier will be trained for every category in M_k. There binary classifiers will be used to classify every documents in $S_j^{'}$. $S_j^{'}$ will be split based on the classification result then. For every category M_{ik} in M_k, documents that will be classified into M_{ik} will form a newly split category. Those documents that can not be classified into any categories in M_k will form another split category. Figure 15 use Venn diagram to illustrate an example of category splitting. Suppose S_1 in the source category tree overlap with two categories in the master category tree M_1 and M_2. Part of the documents in S_1 can be classified into category M_1. This set of documents can be noted as $S_1 \cup M_1$. Another part of the documents in S_1 can be classified into M_2. This set of documents can be noted as $S_1 \cup M_1$. The third part of the documents can not be classified into either M_1 or M_2. This set of documents can be noted as $S_1 - (S_1 \cup M_1) - (S_1 \cup M_2)$.

S_1 should be split into three categories then. They consist of the three parts of documents respectively.

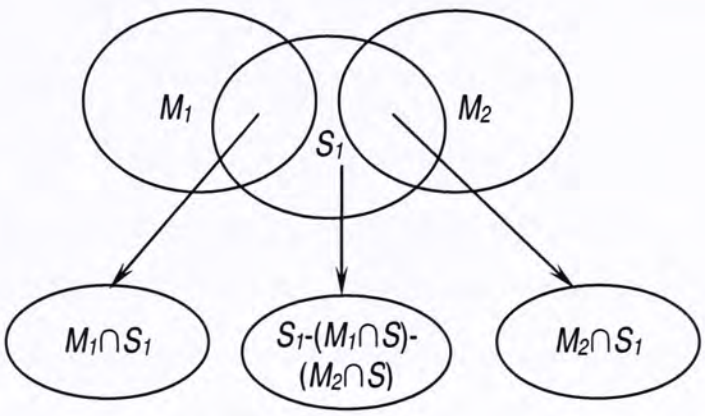


Figure 15. Example of category splitting

We argue that the previous five rules are complete enough for us to deal with all the cases in the integration. As we discussed in the previous section, there are 5 relationships between any two categories in the source and the master category tree. From the set theory, it is obvious that they are also the only five possible relationships. And each of the five rules we develop corresponds to one possible relationship respectively. In Rule 1, S_i *disjoint* with all the categories in the source category tree; in Rule 2, S_i *Match* one of the categories, in Rule 3, S_i *SubConcept* one of the categories; in Rule 4 S_i *SuperConcept* one of the categories; in Rule 5 *Overlap* with one of the categories. Given any category S_i in the source category tree, we can find out a proper rule to integrate it to the proper position. Child-parent relationship, which is the foundation of the rules we developed, has already been embedded into all the rules. Child-parent relationship works in the way of how to integrate the categories in the rules but not the conditions to fire the rules. It can be seen from the rules that how a category is integrated is always based on its parent. All of these rules cover all possible category relationships and all of them require the categories' parent

have a mapping relationship. When we go to the algorithm in the next section, it is found that the roots are initialized to have a mapping relationship.

Rule one to Rule four is not perfect. When a category, say $S_j^{'}$ are inserted as a new category M_{new}, sometimes we don't want to directly insert all the descendants of $S_j^{'}$ as the descendants of M_{new}, because it involves too many categories and once $S_j^{'}$ is mapped to the wrong position, all its descendants will also have a wrong mapping. So, we develop a new rule to make some adjustment in this case. The correct position of $S_j^{'}$ will be further affected by its descendants. Our experiments show that this kind of error often occurs, and this rule is useful.

6) Rule Six : Make Some Adjustment

Given $Insert(S_j^{'}, M_{new}, M_i)$, $S_j = Parent(S_j^{'})$ and $Map(S_j, M_i)$, $M_i^{*} \in Descendant(M_i)$, $S_j^{*} \in Children(S_j^{'})$. To integrate $S_j^{''}$ and $S_j^{'}$, there are three cases to deal with:

Case One: If $Match(S_j^{*}, M_i^{*})$, then $Map(S_j^{''}, M_i^{*})$. If $M_i^{'} = Parent(M_i^{*})$, then $Map(S_j^{'}, M_i^{'})$; else $Insert(S_j^{'}, M_{new}, M_i, M_i^{*})$;

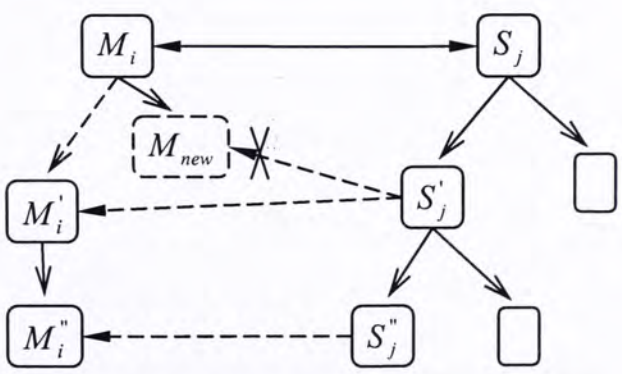


Figure 16. Rule six: Make some adjustment (case one)

Case Two: If $SubConcept(S_j^{"}, M_i^{"})$, then $Insert(S_j^{"}, M_{new}^{'}, M_i^{"})$ and $Map(S_j^{'}, M_i^{"})$;

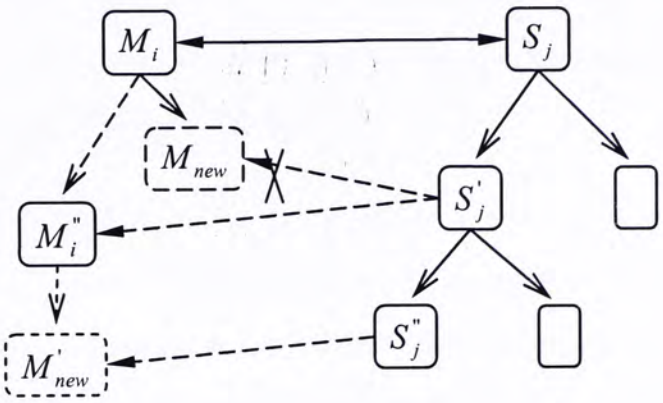


Figure 17. Rule six: Make some adjustment (case two)

Case Three : If $SuperConcept(S_j^{"}, M_i^{"})$, then if $M_i^{'} = Parent(M_i^{"})$, then $Map(S_j^{"}, M_i^{'})$, else $Insert(S_j^{"}, M_{new}^{'}, M_i, M_i^{"})$; if $M_i^{"'} = Parent(M_i^{'})$, then $Map(S_j^{'}, M_i^{"'})$, else $Insert(S_j^{'}, M_{new}^{"}, M_i, M_i^{'})$

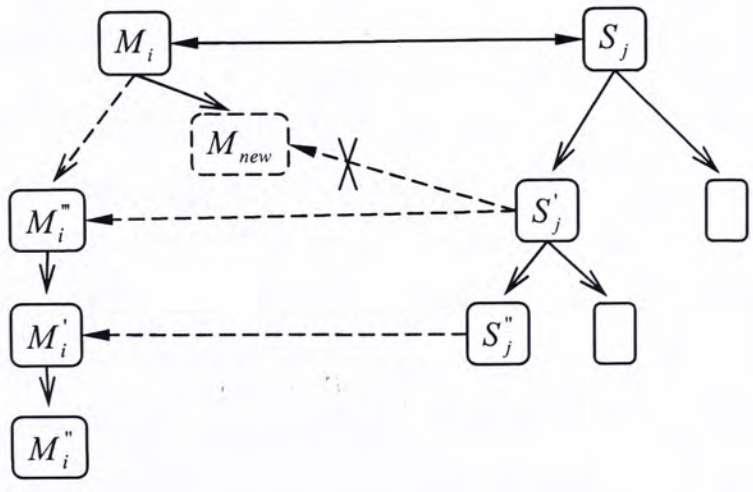


Figure 18. Rule six: Make some adjustment (case three)

Justification: If a category S_j' of the source category tree is added as a new category in the master category tree. But its child S_j'' *match*, *subconcept* or *superconcept* another category in the master category tree within the *Candidate Categories*, then, the position to place S_j' has to be adjusted.

Please note that we discuss "splitting" of a category in the source category tree only, but not the "merging" of two categories in the source category tree. The reason is that the "merging" actually happened in the integration progress, we don't need to develop a specific rule to specify how to merge two categories in the source category tree. For example, if both category *A* and *B* mapped to the same category *C*, they can be regarded as merged together.

4.3. Mapping Algorithm

A mapping algorithm is necessary to clearly describe how to make use the previously described rules and how the whole integration procedure is. We develop a top-down, level-based algorithm. This algorithm tells us when and how to use the decision rules to perform the integration. Source category tree will be mapped to master category tree based on this algorithm.

Basically, we use the breadth-first search algorithm to travel through the source category tree, and dispose all the categories in the source category one by one. Breadth-first search examines all categories connected to the start category before visiting categories further away. Taking the source category tree in Figure 13 as an example, the category "shopping" will be processed first, then comes to "Beauty Products", "Toys and Games", "Health" and "Clothing", the categories "women", "home test" will be processed last. For any given category, say S_j, there are mainly three steps to determine how should we integrate it.

1) Determining Category Relationships

Category relationships between categories in the master and source category tree should be calculated and determined first because all the rules are based on the category relationships. Any well-performed automatic text classifiers can be used to calculate $P(A \mid B)$. For any two categories S_j and M_i in the two category trees, there should be a relationship between them. If we denote the number of categories of source and master categories as $|S|$ and $|M|$ respectively, there will be $|S|*|M|$ relationships in total.

2) Determining Candidate Categories of S_j

S_j has a list category relationships between S_j and all the $|M|$ categories in master category tree, but only part of the categories in the master category tree need to put into consideration as we want to keep the ancestor-descendant relationship between categories. For example, if "Health" in Figure 13 mapped to "Health" in Figure 12, when processing the "Home Test" in Figure 13, we don't need to consider "Personal Care" and "Toys" for any more. This part of categories is named *Candidate Categories* of S_j and denoted as $Candidate(S_j) = \{M_1, ..., M_n\}$. To determine $Candidate(S_j)$, there are several cases:

- If $Map(Parent(S_j), M_i)$, then $Candidate(S_j) = Descendant(M_i)$. This case is actually already embedded in rule one to rule four. It means that if $Parent(S_j)$ mapped to M_i, then S_j can only be mapped to the descendant of M_i, inserted as the descendant of M_i, overlap with the descendant of M_i. If $Split(S_j; S_{j_1}, S_{j_2}..., S_{j_n})$, then $Candidate(S_{ji}) = Candidate(S_j)$, the categories split from S_j share the same candidate categories as S_j.
- If $Split(Parent(S_j); S_{j_1}, S_{j_2}..., S_{j_n})$ and $Parent(S_j)$ overlap the set of categories in $M_k = \{M_{1k}, M_{2k}, ..., M_{nk}\}$ then $Candidate(S_j) = Descendant(M_{1k}) \cup Descendant(M_{2k}) \cup ... \cup Descendant(M_{nk})$. This case deals with the categories

whose parent is split instead of being mapped. It means if $Parent(S_j)$ *overlap* categories in M_k and thus split, then S_j can only mapped to the descendants of categories in M_k, inserted as the descendant of categories M_k, overlap with the descendant of categories in M_k.

- If $Insert(Parent(S_j), M_{new}, M_i)$, then S_j is inserted as the descendant $Parent(S_j)$, no $Candidate(S_j)$ is necessary. This means if $Parent(S_j)$ is inserted as the child of M_i, S_j will be inserted as the child of $Parent(S_j)$ directly.

3) Firing Suitable Rules

If the conditions specified in the rules are satisfied, then the rule will be fired and the S_j will be placed to the proper position in the master category tree. But it is often the case that S_j satisfied the firing conditions of several rules at the same time. S_j may *match* M_1, *subconcept* M_2, *SuperConcept* M_3 and *overlap* M_4 at the same time. But it is obviously more suitable place S_j in only one position in the master category tree instead of making several copies of S_j. So, picking up the proper rule to fire is very important. We give out the priority of all the rules and the order is 1)Rule two, 2)Rule one, 3)Rule three, 4)Rule five and 5)Rule four. Rule six is an adjustment rule, so is not included in this order. If several rules can be fired at the same time, only the rule with highest priority will be fired and others will be omitted.

We argue the priority order can properly deal with all the cases that a category S_j will be encountered. *Disjoint, Match, SubConcept, SuperConcept* and *Overlap* are the only 5 possible relationships between any two categories. So, for S_j and $Candidate(S_j)$, if S_j does not match / subconcept / overlap / superconcept with any of the category of $Candidate(S_j)$, it will disjoint with all of them. Rule two is special because only when S_j disjoint with all the categories in $Candidate(S_j)$, Rule two

will be fired. If any of the category in *Candidate*(S_j) match / subconcept / superconcept / overlap S_j, other rules are more proper. So, Rule two mutual excludes all other rules. And the priority order of Rule one, Rule three, Rule five and Rule four is corresponding to priority order of category relationships *Match, SubConcept, Overlap* and *SuperConcept*. It is obvious that the order of *Match, SubConcept and Overlap* is decided by the category similarity between the two categories involved. *SuperConcept* has the lowest priority because it changes the structure of the master category tree most significantly, which conflicts with our principle of changing the structure of the master category tree as little as possible.

The algorithm we developed in paper is not a symmetric algorithm. For two category trees T^A and T^B, if we set T^A as master category tree and integrate T^B into T^A in case one and set T^B as master category tree and integrate T^A into T^B in case two. The result of the two cases will be different. It makes sense for the algorithm to perform like this because the size and the quality of the category trees are usually different. The result of integrating a small category tree to a large category will be better than the result of integrating a large category tree to a small category tree. And the result of integrating a low quality category to a high quality category will also be better than integrating a high quality category to a low quality category tree. Regarding to the application of category tree integration, the source and master category tree should not be symmetric. In almost all the cases, people would like to integrate many other category trees into his own category which he is familiar with instead of simply integrating two category trees together.

Chapter 5. Experiment of Mono-lingual Category Tree Integration

5.1. Dataset

We collect experiment data from Yahoo! and Open Directory Project (ODP). Yahoo! is one of the most famous directories on the internet. ODP is the largest, most comprehensive human-edited directory of the Web. It is constructed and maintained by a vast, global community of volunteer editors. Netscape Communications Corporation hosts and administers the ODP, and has discretion over its content, use, and operation. Every one who wants to become the editor can apply in its web page, but the quality is still strictly under control, all the application will be evaluated by one of the community's senior editors. "We will make every effort to evaluate all sites submitted to the directory. However, we do not guarantee all submitted sites will get listed. We will be highly selective and judicious about sites we add, and how we organize them." A lot of other search engines use the data of ODP, such as Google, AOL/Netscape, Lycos, and Excite, etc. There are also more and more papers in the field of automated text classification use ODP as their training and testing data.

In the experiment, the data from ODP acts as the source category, and the data from Yahoo! acts as the master category. Every website listed in the corresponding category page is considered as a document in that category. Documents consist of two parts, the website's short description given by editors and its main page. The number of categories and the files are listed in Table 2. The root categories of the master and source category trees map. We didn't use the whole tree of "Science", "Society" or "Shopping", two kinds of processing is applied before running the experiment:

- The height of every category tree in the datasets is 3, because we want to keep the datasets at suitable size for experiment purpose.
- The categories which contain only a few documents are not used, for the classification accuracy will dramatically increase when there is too little training data.

Table 2: The datasets

Master Category Tree			Source Category Tree		
Data Source	**Number of Files**	**Number of Categories**	**Data Source**	**Number of Files**	**Number of Categories**
Yahoo / Science	968	37	ODP / Science	1139	39
Yahoo / Society and Culture	572	25	ODP / Society	993	35
Yahoo / shopping and services	1789	57	ODP / Shopping	3699	68
Yahoo / Recreation / Sports	974	46	ODP / Sports	1476	48
Yahoo / Health	676	33	ODP / Health	1263	37
Average	995.8	39.6	Average	1714	45.4

5.2. Automated Text Classifier

Automated text classifier is required in our experiment to determine category similarity as we have explained in Chapter 4. We use a simple, efficient but still effective method to do the job, the Rocchio method. In Rocchio a class profile is essentially computed as a centroid, a weighted sum of the train documents. It relies on an adaptation to text classification of the well known Rocchio's formula for relevance feedback in the vector-space model. It is first proposed by Hull [16], and has been used by many researchers since then. Rocchio's method computes a classifier $\vec{c}_i = < w_{1i}, ..., w_{|T|i} >$ for category c_i by means of the formula:

$$w_{ki} = \beta \bullet \sum_{\{d_j \in POS_i\}} \frac{w_{kj}}{|POS_i|} - \gamma \bullet \sum_{\{d_j \in NEG_i\}} \frac{w_{kj}}{|NEG_i|} -$$

Where w_{ki} is the weight of t_k in document d_j, $POS_i = \{d_j \in Tr \mid \breve{\Phi}(d_j, c_i) = T\}$, and $NEG_i = \{d_j \in Tr \mid \breve{\Phi}(d_j, c_i) = F\}$. The classifier built by means of the Rocchio method rewards the closeness of a test document to the centroid of the positive training examples, and its distance from the centroid of the negative training examples. In our experiment, β is set to 1 and γ to 0, thus the profile of c_i is the centroid of its positive training examples.

The text classifier introduced above is used for flat classification only. Because we are integrating tree structures, hierarchical classification is necessary. A. Sun et al. [35] proposed a top-down level-based hierarchical approach, and got very good performance. One or more classifiers are constructed at each level of the category tree and each classifier works as a flat classifier at that level. A document will first be classified by the classifier at the root level and then passed into one or more lower level categories. It will then be further classified by the classifiers at the lower level categories until it reaches one or more final categories which could be leaf categories or internal categories. Only binary classifier is trained in this approach. Binary classifiers only make decisions of accepting or rejecting the documents, so a test document may be classified into any number of categories in the taxonomy.

Every category in the hierarchical tree has two classifiers, namely, local classifiers and subtree classifier. Local classifiers determine whether documents should be assigned to the corresponding categories. Subtree classifiers, on the other hand, determine whether documents should be assigned to corresponding category subtrees. For leaf categories, local classifier and subtree classifier are the same. Only if documents are accepted by the subtree classifier, the local classifiers in the work domain of the subtree classifier will have the chance to further classify these

documents. Binary classifiers need to be trained with both positive and negative documents associated with the categories to which the classifiers are assigned. Local and subtree classifiers at different level will have total different positive and negative training data. Let's define some note first. Let $Coverage(C_i)$ be C_i and all the categories in the tree rooted at C_i, $Parent(C_i)$ be the parent category of C_i. The training data of every classifier can be selected as:

- Subtree-classifier for root category C_{root}
  - Positive: All training documents $d_j's$ such that $d_j \in Coverage(C_{root})$
  - Negative: Randomly selected outside the category tree
- Local classifier for root category C_{root}
  - Positive: All training documents $d_j's$ such that $d_j \in C_{root}$
  - Negative: All training documents $d_j's$ such that $d_j \in Coverage(C_{root})$, but $d_j \notin C_{root}$
- Subtree classifier at internal category C_i
  - Positive: All training documents $d_j's$ such that $d_j \in Coverage(C_i)$
  - Negative: All training documents $d_j's$ such that $d_j \in Coverage(Parent(C_i))$, but $d_j \notin Coverage(C_i)$
- Local classifier at internal category C_i
  - Positive: All training documents $d_j's$ such that $d_j \in C_i$
  - Negative: All training documents $d_j's$ such that $d_j \in Coverage(C_i)$, but $d_j \notin C_i$
- Local (Subtree) classifier for leaf category C_l
  - Positive: All training documents $d_j's$ such that $d_j \in C_l$

- Negative: All training documents d_j's such that $d_j \in Coverage(Parent(C_l))$, but $d_j \notin C_l$

5.3. Evaluation Metrics

To have a comprehensive evaluation of the performance of our integration algorithm, we adopt three different evaluation metrics. Each of them evaluates the technique from a different aspect:

- Integration accuracy: It measures the performance of the integration as a whole. We don't use the precision and recall, which is commonly accepted evaluation metric in traditional information retrieval field, because it can not be applied to measure the performance of the algorithm as a whole directly.
- The precision and recall of the three operators. It will measure the performance of the three operators respectively, namely *Map, Insert and Split*. If they are combined together, they can also be used to measure the performance of the whole algorithm.
- The precision and recall of the *Split* operator. We pick *Split* out deliberately because *Split* is a special operator which consists of two steps: 1) deciding if the category should be split. 2) correctly split the categories and new categories are created. This metric evaluate the second step of *Split* only.

The three evaluation metrics will be further discussed in the following sections respectively.

We also manually worked out the correct integrations result discussed in the following sections before the experiment. For each category in the source category, we browse through the content of every document, and then decide the correct position the master categories it should be integrated to. The datasets we collect are very clear, which means we can easily tell the correct integration position of the categories in most cases. But, some times it seems there are multiple positions that a specific category can be integrated to. In this case, we will decide the most proper

position based on the real situation. For example, "Asian Literature" in the source category tree can be both the parent category of "Chinese Literature" in the master category tree or it can also be split into two or more categories. If the documents about "Chinese Literature" take up a large proportion in the "Asian Literature", I will prefer to split it and if the document about "Chinese Literature" takes up just as much as "Japanese Literature", "Korean Literature", "Iran Literature" etc., I will prefer to added it as the parent of "Chinese Literature".

5.3.1. Integration Accuracy

We can measure the accuracy of the integration in two different levels. The first one is the integration accuracy of categories, which evaluate how correctly the categories are integrated by the algorithm. It is measured by the percentage of categories that are placed in the correct positions. The second one is the integration accuracy of documents, which evaluate how correctly the documents are integrated by the algorithm. It is measured by the percentage of documents which are placed in the correct positions. They can be calculated in the following formula:

$$Accuracy_{category} = \frac{\text{number of correctly integrated categories}}{\text{number of categories in the source category tree}},$$

$$Accuracy_{doc} = \frac{\text{number of correctly integrated documents}}{\text{number of documents in the source category tree}}$$

The $Accuracy_{doc}$ can be different from $Accuracy_{category}$ because part of the wrongly split categories may be mapped to the right position later. Not all the documents belonged to the wrongly processed categories are placed in the wrong position. Another reason to differentiate $Accuracy_{doc}$ and $Accuracy_{category}$ is that the number of documents in each category can be different. It is also a factor that will affect the value of the $Accuracy_{doc}$ even if $Accuracy_{category}$ is the same.

5.3.2. *Precision and Recall and F_1 value of the Three Operators*

We also evaluate the performance of three integration operators in our experiment. Precision and recall are used to describe them. Precision and recall are defined as follows:

$$Precison = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}$$

TP: the number of categories that are correctly mapped, inserted or split. Note that for *Map* and *Insert*, the categories should not only be mapped or inserted, but also be mapped or inserted to the right position; for *Split*, how the categories are split is not put into consideration in this metric. The case of *Split* is a little different from the other two operators and will be discuss in Section 5.3.3

FP: the number of categories that are wrongly mapped, inserted or split.

FN: the number of categories that are should be mapped, inserted or split, but not.

However, neither Precision nor Recall alone can accurately assess the quality. In particular, Recall and easily be maximized at the expense of a poor precision. A high Precision can be achieved at the expense of a poor recall. Hence, it is necessary to consider a combined measure, we use the F_1 value:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

5.3.3. *Precision and Recalls of "Split"*

The operator *Split* is a little different from *Map* and *Insert* for it involves the original category being split and the categories split from the operator and consists of two steps:

- Determine if a category should be split or not. How to evaluate this step has already been discussed in Section 5.3.2. It involves the original category being split only.
- Determine how to split the categories. This is what we will discuss here. It involves the categories we get after the splitting.

We use precision, recall and F_1 again to describe the performance, they are defined as follows:

$$Precison_{split} = \frac{TP_{split}}{TP_{split} + FP_{split}}, \quad Recall_{split} = \frac{TP_{split}}{TP_{split} + FN_{split}}$$

$$F_{1,split} = 2 * \frac{Precision_{split} * Recall_{split}}{Precision_{split} + Recall_{split}}$$

TP_{split} : the number of new categories are correctly created.

FP_{split} : the number of categories that should not be created.

FN_{split} : the number of categories that should be created but not.

5.4. Parameter Turning

Category relationships play a very important role in the integration process. Because of the classification error, we use th_H and th_L instead of 1 and 0 to determine them as discussed previously. Selecting proper values of the two parameters have a significant impact in our algorithm. So, we have to tune the parameters to get the best result before the experiment. The two parameters are tuned respectively, because they are independent to each other. The footstep we choose for the tuning is 0.005, which is small enough in our experiment.

Please refer to Figure 19, Figure 20 and Figure 21 for the F_1, precision and recall of the three operators and the micro-average value of these operators respectively when we are tuning th_H with th_L set as 0.3. Theoretically, th_H should be tuned in the range [0.5, 1.0], but the figures give the result between [0.5, 0.9] only, because when th_H is larger than 0.9, we have already seen a significant decline of the performance.

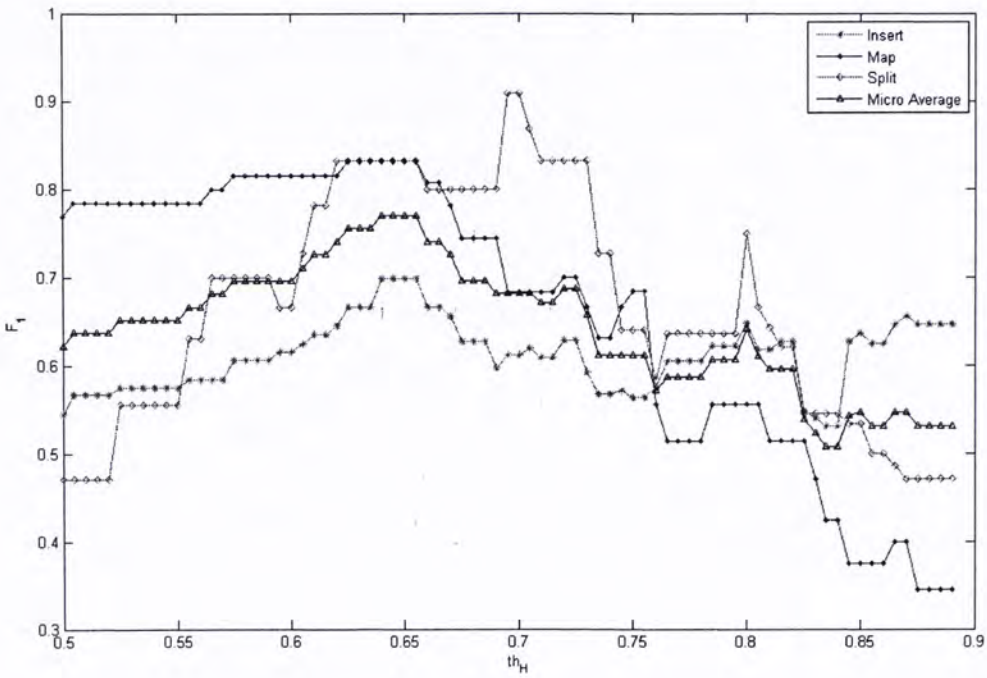


Figure 19. F_1 value when tuning th_H

Roughly, the main trend is that as the th_H increases, the F_1 value of *Insert, Map, Split* and the *Micro Average* of them increase at the beginning and reach the peak at some specific values in the middle. After these peak values, the F_1 values decrease as the th_H increase. The lines seems not very continues, because how to integrate every single category is closely related to its parent/child/sibling. The way to integrate an important category will affect the way to integrate many related categories. For

example, if a category is mapped to a wrong position in the master category, all its children may also be placed to the wrong positions.

Figure 19 shows that in the range [0.64, 0.655], the F_1 value of micro-average gets the best result. *Insert, Map, Split* get to the peak when the th_H is in the range of [0.64, 0.655], [0.625, 0.655] and [0.62, 0.655] respectively. We take th_H as 0.65 in our experiment since it is in the middle of the values that make the *Micro Average* of F_1 of the three operators get the best result.

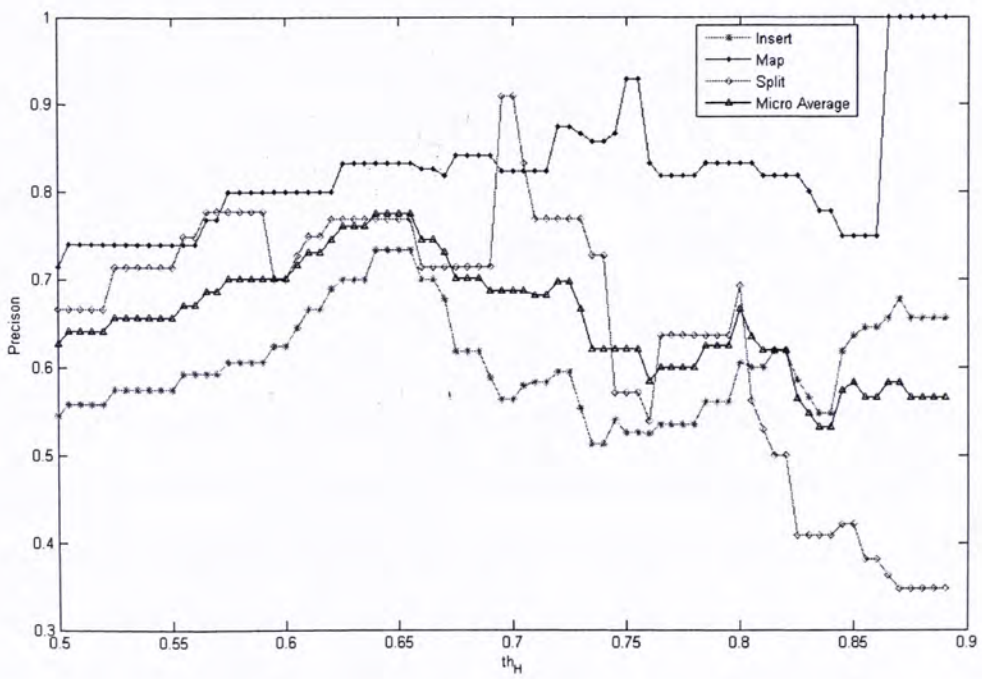


Figure 20. Precision when tuning th_H

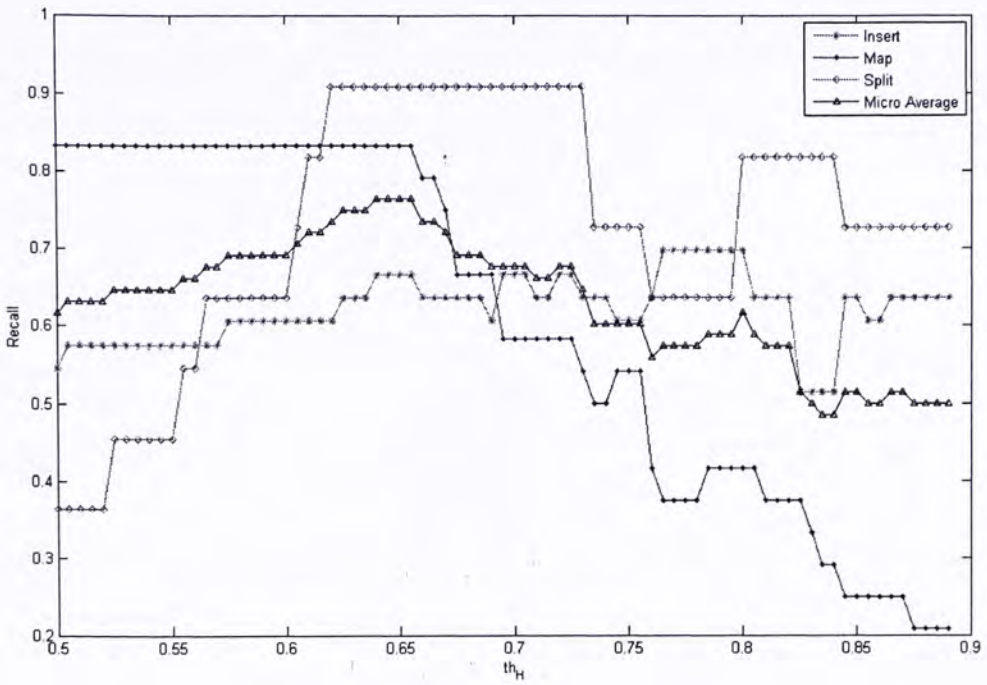


Figure 21. Recall when tuning th_H

Figure 20 and Figure 21 show the precision and recall of the operators when we are tuning th_H . *Map* seems very special. The precision of operator *Map* raises to 1 when th_H equals to 0.865 and the recall of operator *Map* falls to 0 when th_H equals to 0.89. The reason is that operator *Map* is closely related to category relationship *Match*, which is solely affected by th_H . High th_H makes the judgment of *Match* relationship more accurate, thus the precision increase as the th_H increase. When th_H equals to 1, no category still match another other categories, thus the number of *Map* decreases to 0, the recall decreases to 0 too. The precision and recall of other operators and micro average of these operators are similar to F_1 value. The precision and recall rise at the beginning as the th_H increase and drop after the specific peak values.

Please refer to Figure 22, Figure 23 and Figure 24 for the F_1, precision and recall of the three operators and the micro-average of the operators respectively when we are tuning th_L with th_H set as 0.65.

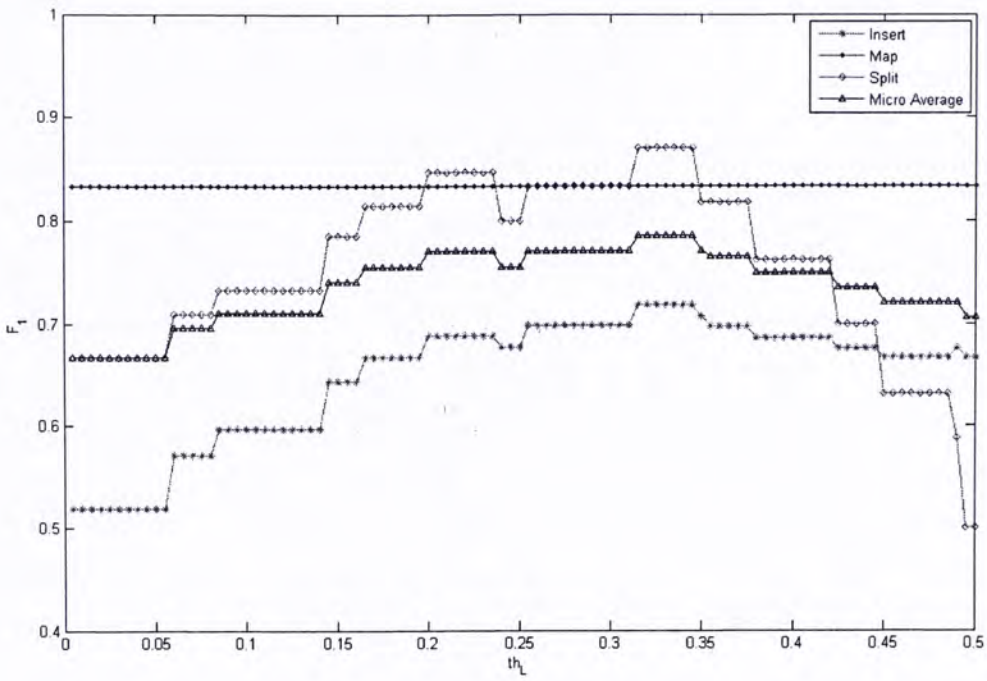


Figure 22. F_1 value when tuning th_L

It can be seen from Figure 22 that in the range [0.315, 0.345], the F_1 value of micro-average get the best result. So, we take th_L as 0.33 in our experiment since it is in the middle of [0.315, 0.345]. The trend of this figure is also that as the th_L increases, the F_1 values of *Insert*, *Split* and *Micro Average* increase at the beginning and reach the peak at some specific values. After these specific values, the F_1 values decrease as the th_L increase. *Insert* and *Split* reach the peak when the th_L is in the range of [0.315, 0.345] and [0.315, 0.345] respectively. Map is very special again. The

performance of operator *Map* remains constant as 0.833 since it is affected by th_H only. If th_H doesn't fluctuate, neither does the performance of *Map*. Of course, the precision and recall of *Map* remain constant in Figure 23 and Figure 24 too.

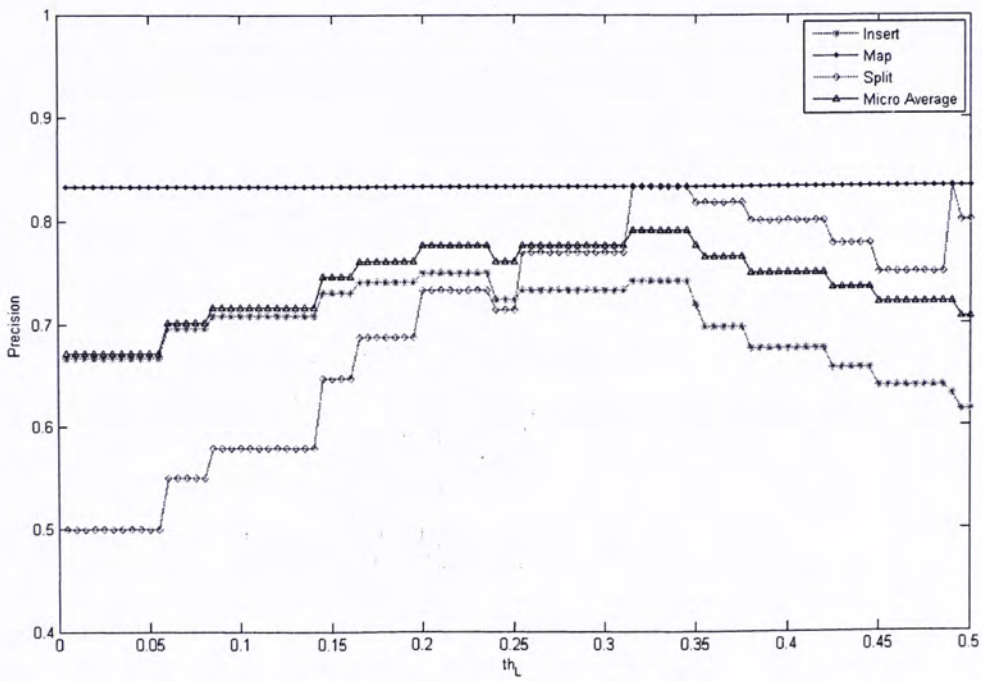


Figure 23. Precision when tuning th_L

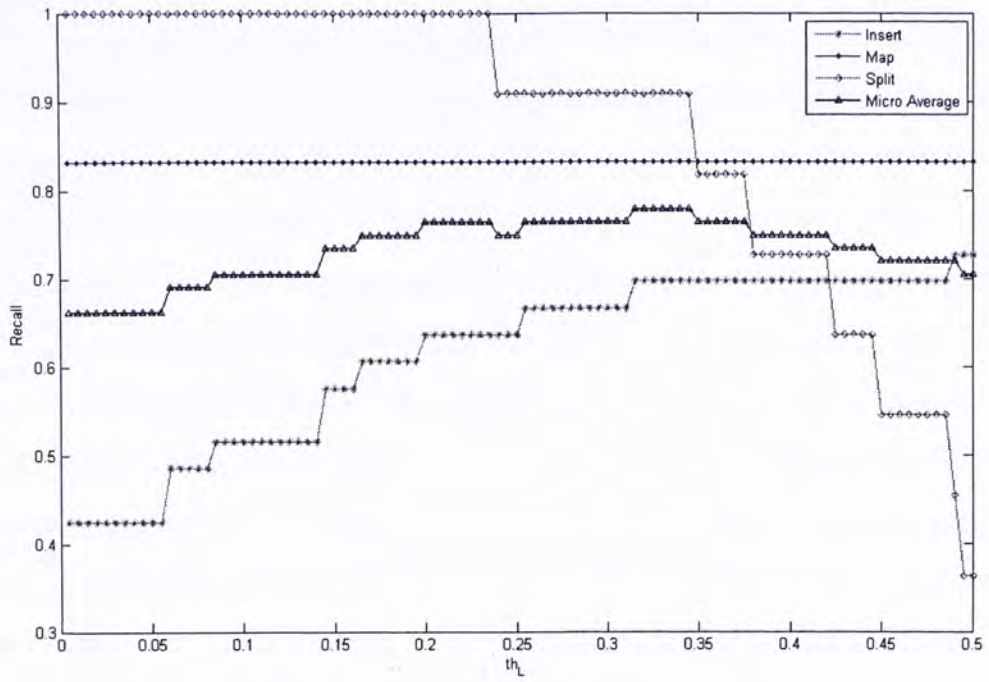


Figure 24. Recall when tuning th_L

Figure 23 is similar to the previous several figures, as the increase of th_L, the *Insert*, *Split* and *Micro Average* increase at the beginning and decrease after the peak values. Figure 24 is special, since the recall of *Split* decreases continually and the *Insert* increases continually. The reason is that as the th_L approaches 0, the number of categories relationship judged as *Overlap* increase continually. Thus, the number of categories split increase continually and the number of categories inserted decrease continually. Accordingly, the recall of *Split* and *Insert* decrease or increase continually.

5.5. Experiments Results

Table 3 shows the integration accuracy and some other statistical data of the experiment. The average integration accuracy of categories and documents are more

than 81.63% and 80.60% respectively. This is a good performance. For how the categories are integrated, even if the master and the source category tree are rooted at the same category, the number of categories inserted still takes up the most proportion of all the categories, which means the similarity between the master and the source category tree is not very high. Different people may pay more attention to different contents when building category trees of the same topic. By integrating multiple category trees together, the users can get a much more comprehensive one. The number of categories split takes the least proportion of the all categories. Not many categories between the master and source category trees overlap with each other. The trees share the same or very similar principle to further divide large categories. For example, in both Yahoo! and ODP, the "Science" is further divided by discipline. They both have the children categories like "Physics", "Chemistry", "Math" et al. Please also note that there are several empty categories which do not have any documents belong to them. They are usually the internal categories and are not integrated in our algorithm. Their number is shown as "Number of Empty Categories" in the table.

Table 3: Category tree integration result I

	Science	**Society**	**Shopping**	**Sports**	**Health**	**Average**
$Accuracy_{category}$	79.49%	82.35%	77.94%	79.17%	89.19%	81.63%
$Accuracy_{doc}$	79.89%	69.48%	78.32%	83.37%	91.92%	80.60%
Number of Categories Split	8	4	12	6	2	6.4
Number of Categories Mapped	18	12	24	20	15	17.8
Number of Categories Inserted	13	18	31	21	20	20.6
Number of Empty Categories	0	1	1	1	0	0.6

The recall and precision of the three operators are shown in Table 4. The F_1 values of the three operators *Split*, *Map* and *Insert* are 79.20%, 86.43% and 76.91%

respectively. The performance of *Map* is much better than that of the other two operators. This is because of the relative importance of the three operators. It can be found that *Map* is the key of the three operators if the rules and the algorithm are examined more carefully. The errors of *Split* and *Insert* come from not only the two operators themselves, but also *Map*. They also suffer part of errors made by *Map*, because if a category is not mapped to the correct position it should be mapped, there is little chance that its children can be correctly processed. For *Split*, even if a node is not correctly split, part of its children still have the chance to be correctly processed. For *Insert*, we have Rule six to deal with this problem. Our tuning method is based on the *Micro Average* of F_1 of the three operators, which evaluate the whole performance of the three operators.

Table 4: Category tree integration result II

	Split			*Map*			*Insert*		
	Precision	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**
Science	87.50%	77.78%	82.35%	94.44%	85.00%	89.47%	53.85%	77.78%	63.64%
Society	75.00%	60.00%	66.67%	83.33%	83.33%	83.33%	83.33%	88.24%	85.71%
Shopping	83.33%	90.91%	86.96%	83.33%	83.33%	83.33%	74.19%	69.70%	71.87%
Sports	66.67%	100.0%	80.00%	90.00%	81.81%	85.71%	71.43%	71.43%	71.43%
Health	100.0%	66.67%	80.00%	93.33%	87.50%	90.32%	85.00%	100.0%	91.89%
Average	82.50%	79.07%	79.20%	88.89%	84.19%	86.43%	73.56%	81.43%	76.91%

Please refer to Table 5 for how Rule 2, Rule 3 and Rule 5 affect *Insert* in our experiment. Table 6 also shows how frequent the three rules are fired. There are several N/A of Rule 4 in the form because Rule 4 is not applicable in the datasets *Science* and *Sports*. It is never and also should never be fired in the two datasets. The average F_1 value of Rule 4 is much better than the average F_1 value of *Insert,* but for Rule 4 is not fired very frequent, so it will not change the integration performance of

Insert significantly. The average F_1 value of Rule 2 and Rule 3 are close to the average F_1 value of *Insert* which means they make similar contribution to the overall integration performance of *Insert*.

Table 5. Category tree integration result III

	Rule 2			*Rule 3*			*Rule 4*		
	Precision	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**
Science	50.00%	83.33%	62.50%	66.67%	66.67%	66.67%	N/A	N/A	N/A
Society	85.71%	100.0%	92.31%	75.00%	60.00%	66.67%	100.0%	100.0%	100.0%
Shopping	75.00%	69.23%	72.00%	71.43%	71.43%	71.43%	66.67%	66.67%	66.67%
Sports	64.71%	68.75%	66.67%	100.0%	80.00%	88.89%	N/A	N/A	N/A
Health	87.50%	100.0%	93.33%	75.00%	100.0%	85.71%	100.0%	100.0%	100.0%
Average	72.58%	84.26%	77.36%	77.62%	75.62%	75.87%	88.89%	88.89%	88.89%

Table 6. Frequency of the rules being fired

	Rule 2	*Rule 3*	*Rule 4*
Science	10	3	0
Society	14	4	1
Shopping	17	4	0
Sports	24	7	3
Health	16	4	1
Average	16.2	4.4	1

From Table 7 we can see the performance of the second step of *Split* operator. It gets an average F_1 value of 80.11%. This number is also acceptable. But the precision and recall in this table vary significantly. Large gaps exist between the numbers. It can be seen from Table 3 that the average number of categories split in the source category tree is 6.4. Even if one or two categories are wrongly split, the performance will greatly deteriorate.

Table 7: Category tree integration result IV

	Precision	*Recall*	F_1
Science	83.33%	77.78%	80.45%
Society	75.00%	60.00%	66.67%
Shopping	85.19%	80.00%	82.51%
Sports	83.33%	100.0%	90.91%
Health	100.0%	66.67%	80.00%
Average	85.37%	76.89%	80.11%

Chapter 6. Cross-lingual Category Tree Integration

In this chapter, we extend our existing mono-lingual category integration technique proposed in Chapter 4 with cross-lingual semantic interoperability. Please refer to Figure 25 to the overall process of cross-lingual category tree integration. We first build a cross-lingual concept space by doing some statistical analysis on a parallel corpus. And then both the source and the master category tree are mapped to the cross-lingual concept space. The category integration algorithm will be applied to the processed category trees to integrate them together.

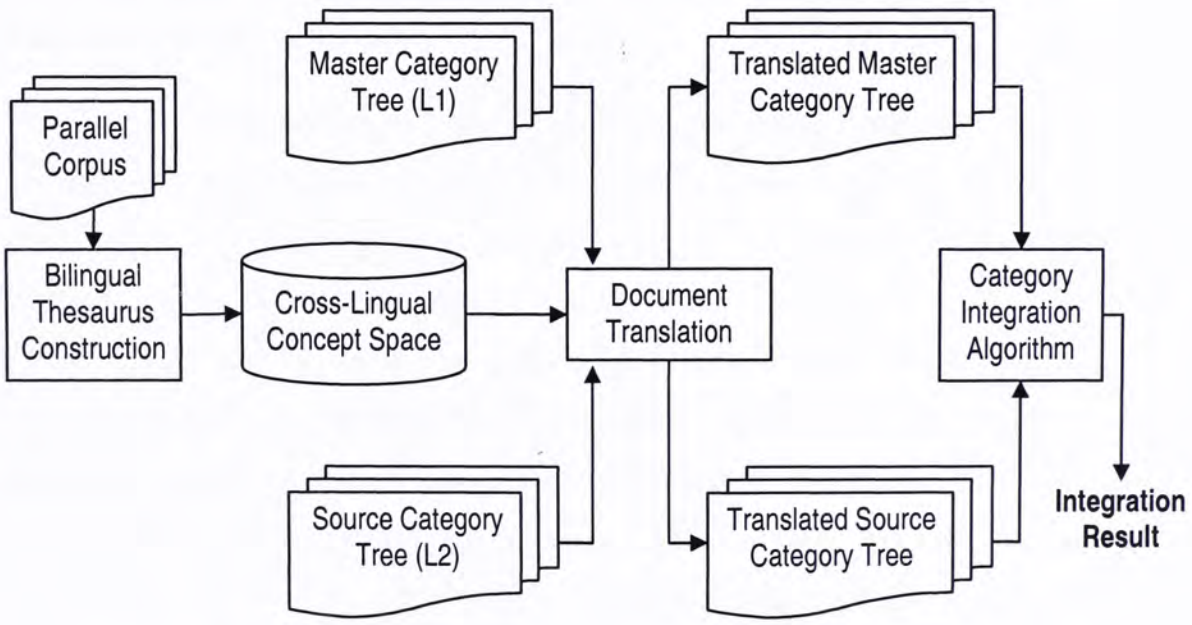


Figure 25. Overall process of cross-lingual category tree integration

6.1. Parallel Corpus

Cross-lingual semantic interoperability is a challenge in multilingual knowledge management systems. Dictionary is a tool that is widely utilized in commercial systems to cross the language barrier. However, terms available in dictionary are always limited. As language is evolving, there are new words being created from time to time. For example, there are new technical terms and named entities such as RFID and Baidu. To solve the problem of cross-lingual semantic interoperability, corpus-based approach is better than the controlled vocabulary approach and the knowledge-based approach. It makes use of the statistical information of term usage in a multilingual corpus to automatically construct a statistically base cross-lingual thesaurus to overcome the limitation of knowledge-based approach. Multilingual corpus is a collection of text in electronic form (written language corpus) where texts in different languages are put together either based on parallelism or comparability. It is necessary in corpus approach.

Multilingual corpus constructed based on comparability and parallelism is known as comparable corpus and parallel corpus respectively. Comparable corpus is defined as a collection of texts composed independently in the respective languages and combined on the basis of similarity of content, domain and communicative function. Parallel corpus can be developed using overt translation or covert translation. The overt translation posses a directional relationship between the pair of texts in two languages, which means texts in language *A* (source text) is translated into texts in language *B* (translated text). The covert translation is non-directional. Multilingual documents expressing the same content in different languages are generated by the same source. Therefore, none of the text in each pair of such parallel corpus is marked as translated text or source text.

Although the availability of comparable corpus is higher than the availability of parallel corpus, it is difficult to justify the criteria for constructing comparable corpora. In addition, it is complicated to generate the hypotheses of possible

alignments. Greater Certainty as to the equivalence of particular expressions can be obtained by using parallel corpora. Hong Kong is a good place to collected parallel corpus, for both English and Chinese is the official language in Hong Kong. Many Web sites hosted in Hong Kong provide documents in both English and Chinese. Due to the special colonial history, some of the legal documents are even required to provide both versions by law. For example, the Hong Kong Special Administrative Region (HKSAR) Government has published many governmental, legal and financial documents on the Web for public access, such as press release articles. These documents are written in both Chinese and English based on covert translation. The bilingual documents are organized on the Web site using the mono-lingual subtree structure where there are no direct links between the English and Chinese documents. C. C. Yang and K. W. Li [39] have developed a title English/Chinese alignment model to automatically construct English / Chinese Parallel Corpus from these sources.

This alignment model is based on the title of the documents. To overcome the lexical and grammatical problems, the longest common subsequence, employed in sequence comparison methods, is utilized to optimized the alignment of English and Chinese titles. It consists of three major steps:

1) Alignment at word level and character level

An English title, E, is formed by a sequence of English simple words, i.e., $E = e_1 e_2 e_3 ... e_i ...$, where e_i is the i^{th} English word in E. A Chinese title, C, is formed by a sequence of Chinese characters, i.e., $C = char_1 char_2 char_3 ... char_q ...$, where $char_q$ is the q^{th} Chinese character in C. An English word in E, e_i, can be translated to a set of possible Chinese translations, $Translated(e_i)$, by dictionary lookup. $Translated(e_i) = \{T_{e_i}^1, T_{e_i}^2, T_{e_i}^3, ..., T_{e_i}^j, ...\}$, where is the $T_{e_i}^j$ is the j^{th} Chinese translation of e_i. A sequence of Chinese characters forms each Chinese translation. The set of

the LCSs of a Chinese translation $T_{e_i}^j$ and C is $LCS(T_{e_i}^j, C)$. $MatchList(e_i)$ is a set that holds all the unique LCSs of $T_{e_i}^j$ and C for all Chinese translations of e_i

$$MatchList(e_i) = \bigcup_j LCS(T_{e_i}^j, C).$$

If there is no common subsequence of $T_{e_i}^j$ and C, then $MatchList(e_i) = \varnothing$ and no reliable translation of e_i can be found in C.

Assume that if the characters of the Chinese translation of an English word appear adjacently in a Chinese sentence, then the Chinese translation would be more reliable than translations in which the characters do not appear adjacently in the Chinese sentence. This hypothesis is thus applied to the algorithm. $Contiguous(e_i)$ is used to determine the most reliable translation based on adjacency.

$$Contiguous(e_i) = \{x \mid x \in MatchList(e_i) \text{ and all the characters of } x \text{ that appear adjacently in } C\}$$

The second criterion for the most reliable Chinese translation is the length of the translation. $Reliable(e_i)$ is used to identify the longest sequence in $Contiguous(e_i)$

$$Reliable(e_i) = \begin{cases} \arg\max_{x \in Contiguous(e_i)} |x| & \text{if } Contiguous(e_i) \neq \varnothing \\ \arg\max_{x \in MatchList(e_i)} |x| & \text{otherwise} \end{cases}$$

2) Resolving redundancy

Redundancy has the primary function of adding cohesion in a language but at the same time, it is a problem for alignment. Because of redundancy, the translations of an English word may overlap partially or completely in Chinese. To deal with redundancy, $Dele(x, y)$ is added as an edit operation to remove the $LCS(x, y)$ from x.

WaitList is a list that saves all of the sequences that are obtained by removing the overlapping of the elements of $MatchList(e_i)$ and $Reliable(e_i)$.

$$WaitList = DELE(WaitList, Reliable(e_i)) \cup DELE(MatchList(e_i) \setminus \{Reliable(e_i)\}, Reliable(e_i)),$$

Where $DELE(X, y) = \bigcup_{i=1}^{n} Dele(x_i, y)$ and x_i is the i^{th} element of X

3) Alignment at title level

Given E and C, the ratio of matching is determined by the portion of C that matches with the reliable translations of English words in E. *Remain* is a sequence that is initialized as C, and $Reliable(e_i)$ is removed from *Remain* from the e_i until the last English word.

$$Matching\_Ratio(E, C) = \frac{|C| - |Remain|}{|C|}$$

For any given English title, the Chinese title that has the highest $Matching\_Ratio$ among all of the Chinese titles is considered to be the counterpart of the English title. If more than one Chinese title has the highest $Matching\_Ratio$ to the English title E, then the Chinese title with the lowest value of $|Matching\_Ratio(E, C) - Matching\_Ratio^*(E, C)|$ is considered to be the counterpart of E.

$$Matching\_Ratio^*(E, C) = \frac{\sum_i R(e_i)}{|E|}$$

Where $R(e_i) = \begin{cases} 0 & if\ Reliable(e_i) = e \\ 1 & otherwise \end{cases}$

For the parallel corpus we used in the cross-lingual category tree integration experiment, 2632 documents pairs are collected from HKSAR Government's website. These documents are subjected in many different areas, including commerce, culture, information technology, health, security et al.

6.2. Cross-lingual Concept Space Construction

Cross-lingual concept space and a bilingual thesaurus will be constructed from these parallel corpora. This is the heart of the cross-lingual category tree integration, because it solves the cross-lingual semantic interoperability issue. C.C. Yang et al [41][45] propose a associate constraint network approach to construct cross-lingual thesaurus. There are mainly three components for the cross-lingual concept space construction, namely, phase extraction, co-occurrence analysis, and concept space generation.

6.2.1. Phase Extraction

The phrase extraction of the English and Chinese documents identifies important conceptual phrases in the corpus. In the English phrase extraction, a stop-word list and term-phrase formation are utilized. A stop-word list is a list of non-semantic bearing words such as 'the', 'a', 'on' and 'in'. The stop words are first removed from the English documents, and the term-phrase formation is then utilized to formulate phrases by combining adjacent words. For example, 'international crime' is a term phrase that is formed by the two adjacent words, 'international' and 'crime'. In the Chinese phrase extraction, some Chinese text segmentation techniques such as the one proposed in [38] should be used. It is based on mutual information, and the significant estimation of adjacent Chinese characters.

6.2.2. Co-occurrence analysis

In the Chinese/English parallel corpus, N pairs of Chinese documents and English documents, C_i and and E_i (i=1,2,…,N), are aligned. For each document pair, the M

most significant Chinese terms and the M most significant English terms are extracted based on the term weights (d_{ij} and d_{ij^*}) that are computed by the term frequencies and the inverse document frequencies. The term frequency factor provides a measurement of how well a term describes the document contents, which is called the intra document characterization. The inverse document frequency factor is a measurement of inter-cluster similarity, which is important because terms that appear in many documents are not useful in distinguishing a relevant document from a non-relevant document. A log function is typically applied to the inverse document frequencies to penalize terms that appear in many documents. This function has been proved to be effective, and has been applied in most information retrieval techniques.

$$d_{ij} = tf_{ij} \times \log(\frac{N}{df_j} \times w_j)$$

$$d_{ij^*} = tf_{ij^*} \times \log(\frac{N}{df_{j^*}} \times w_{j^*})$$

The length of an English term is determined by the number of words, and the length of a Chinese term is determined by the number of characters.

After extracting the most significant terms from N document pairs, the relevance weights between the extracted terms are computed based on the co-occurrence analysis. The co-importance weight d_{ijk} between term j and term k is computed as follows, where term j and term k can be Chinese terms or English terms (term j^* and term k^*)

$$d_{ijk} = tf_{ijk} \times \log(\frac{N}{df_{jk}} \times w_j)$$

Where tf_{ijk} is the minimum of tf_{ij} and tf_{ik} in document pair i and df_{jk} is the document frequency of both term j and term k.

d_{ijk^*} corresponds to the co-importance weight between Chinese term j and English term k^* in document pair i, d_{jik} corresponds to the co-importance weight between Chinese term j and Chinese term k and $d_{ij^*k^*}$ corresponds to the co-importance weight between English term j^* and English term k^*.

The relevance weights between term j and term k, W_{jk} and W_{kj}, are then computed based on the following asymmetric functions.

$$W_{jk} = \frac{\sum_{i=1}^{N} d_{ijk}}{\sum_{i=1}^{N} d_{ij}} \times WeightFactor(k)$$

$WeightFactor(k) = \frac{\log \frac{N}{df_k}}{\log N}$. The relevance weights between term j and term k are asymmetric. If term j is more significant than term k ($\sum d_{ij} > \sum d_{ik}$), then W_{jk} is less than W_{kj}, which means that the more significant term will have less impact on the less significant term. For example, the term 'peer to peer' has less impact on the term 'Internet', and thus 'Internet' will not be included in the concept space of 'peer to peer'.

6.2.3. Associate Constraint Network for Concept Generation

Associate constraint network approach is utilized to construct a cross-lingual concept space. The cross-lingual concept space is modeled as an associate constraint network, and the problem of generating the cross-lingual concept space is formulated as a constraint satisfaction problem.

A constrain satisfaction problem (CSP) is a problem composed of a finite set of variables, each of which is associated with a finite domain, and a set of constraints that restricts the values the variables can simultaneously take. The task is to assign a value to each variable satisfying all the constraints. In this problem, the nodes of an associate constraint network $(x_1, x_2, ..., x_n)$ represent the extracted terms of the parallel corpus, where x_i can be a term in L_1 and L_2. The values of the nodes are binary, x_j = {0,1}:

- x_j=1, if x_j is a term in the cross-lingual concept space and
- x_j=0, if x_j is not a term in the cross-lingual concept space

The arcs of the associate network represent the association between the extracted terms. The constraint c_j is applied on x_j. A term, x_j, is considered to be relevant to other terms in the thesaurus if the sum of the associate weights between the other terms and itself is sufficiently large; otherwise, it should not be included in the thesaurus. c_j is given as below.

$$x_j = \begin{cases} 1 & \sum_{i=1,i\neq j}^{n} W_{ij} x_i \geq threshold \\ 0 & \sum_{i=1,i\neq j}^{n} W_{ij} x_i < threshold \end{cases}$$

The constraint satisfaction problem (CSP) for constructing a cross-lingual thesaurus is then defined in terms of the node consistency and the satisfaction of the association constrain network as follows:

Node Consistency:

x_j is consistent if and only if c_j is satisfied in the associated constraint network.

Associate Constraint Network Satisfaction:

The associate constraint network is satisfied if and only if all nodes in the associate constraint network are consistent and $\sum_j x_j < C$, where C is a threshold and determined statistically based on the distribution of an input concept and their associated concepts.

A solution to a CSP is an assignment of a value from its domain to every variable. There are basically two search strategies in associate constraint network.

- Backtracking scheme
- Look ahead scheme

For more detailed about the two search strategies, please refer to [45].

6.3. Document Translation

After the cross-lingual concept space and bi-lingual thesaurus are constructed, both the source category tree and the master category tree needed be translated into the new concept space by using this thesaurus.

Documents cannot be directly interpreted by the computer algorithm. So, an indexing procedure that maps a document d_j into a compact representation of its content needs to be applied at the very beginning of the algorithm. The representation of the documents depends on the concept space applied to the documents. For the cross-lingual category tree problem, we have two concept spaces at first, concept space in Chinese and concept space in English. Documents written in Chinese or English are represented in the two concept space respectively. The $TF \times IDF$ scheme is employed to represent each document d_j in the k dimensional space and form a document feature vector $\vec{d}_j = < w_{1j}, w_{2j}, ..., w_{|L|j} >$ in Chinese or in English. $|L|$ is the number of concepts in the Chinese or English concept space. But, Section 6.2 builds a new unified cross-lingual concept space based on both the Chinese and the English concept spaces. The terms in the cross-lingual concept space can be Chinese or

English, and these terms are extracted from the original Chinese or English concept space by the associate constraint network. The bi-lingual thesaurus keeps the record of the relevance weights between different terms in the cross-lingual concept space. The task of document translation is to project all the document vectors represented in the Chinese concept space or in the English concepts space into the cross-lingual concept space.

Let's denote the category tree before the mapping as $T = \{C, E\}$. Because the translation is at the document level, we didn't specify the language of the category tree. This notation can be used in all the three cases:

- Different documents within a category use different language;
- Different categories use different languages, but the language use within a category is the same;
- Different category trees use different languages, but the language used within a category tree is the same

The structure of T will remain the same. So, nothing needed to be changed in E and the number and positions of categories in C will also remain unchanged. But every document represented in a vector $\vec{d}_j$ in Chinese or English concept space in all the categories should be replaced by a new vector $\vec{d}_j^{new} = < w_{1j}^{new}, w_{2j}^{new}, w_{3j}^{new}, ..., w_{|T|^{new}j}^{new} >$ represented in the cross-lingual concept space, where w_{ij}^{new} is the weight of document j on the i^{th} dimension.

The translation is based on the bi-lingual thesaurus which keeps the record of the relevance weights between different terms in the cross-lingual concept space. The thesaurus keeps not only the relevance weights between Chinese terms and English terms, but also the relevance weights between terms of the same language. For example, the relevance weights between term "中藥" (Traditional Chinese Herbal Drug, p) and term "中醫" (Traditional Chinese Medicine, q) are 0.77 (W_{pq}) and 0.52

(W_{qp}); the relevance weights between term "postcode"(p) and term "postal" (q) are 1.00 (W_{pq}) and 0.25 (W_{qp}).

Any term in the cross-lingual concept will have a set of relevant terms, let's denote it as $N_{term_i} = \{term_1, term_2, ...\}$. $term_i$ itself is not included in N_{term_i}. In order to project $\vec{d}_j$ into $\vec{d}_j^{new}$, we have to determine the weight of every component in $\vec{d}_j^{new}$. For $term_i$ in the cross-lingual concept space, its weight can be calculated by summing up the weights of all the elements in the set N_{term_i} in $\vec{d}_j$ multiplied by the relevance weights between $term_i$ and the element terms. Thus, any component w_{ij}^{new} in the new vector $\vec{d}_j^{new} = < w_{1j}^{new}, w_{2j}^{new}, w_{3j}^{new}, ..., w_{|T^{new}|j}^{new} >$ can be determined by the following formula:

$$w_{ij}^{new} = \sum_{k=1}^{|N_{term_i}|} w_k \times W_k + w_i$$

Where w_k is the weight of relevant $term_k$ in $\vec{d}_j$, w_i is the weight of $term_i$ in $\vec{d}_j$ and W_k is relevance weights between $term_k$, and $term_i$. By calculating the components of $\vec{d}_j^{new}$ one by one, the new vector can be translated from $\vec{d}_j$ directly.

After the vector projection phase, all the vectors in both Chinese and English concept are projected to the same uniform semantic space. Associated terms in different languages can be interpreted consistently by the integration algorithm. Let us suppose a Chinese document is represented in a Chinese vector space with only two terms $\{m,p\}$ as $\vec{d}_c = \{0.65, 0.75\}$ and an English document in an English vector space with term $\{n,q\}$ as $\vec{d}_e = \{0.6, 0.8\}$. The cross-lingual concept space consist of all the four terms $\{m, p, n, q\}$ with W_{mn}=0.85, W_{nm}= 0.9, W_{pq}= 0.2, W_{qp}=0.3. Then, the Chinese vector can be projected as $\vec{d}_c^{new}$ =$\{0.65, 0.75, 0.65*0.85, 0.75*0.2\}$=$\{0.65, 0.75,$

0.5525, 0.15} and the English vector can be projected as $\vec{d}_e^{new}$ *={0.6*0.9, 0.8*0.3, 0.6, 0.8}={0.54, 0.24, 0.6, 0.8}*. The documents $\vec{d}_c^{new}$ and $\vec{d}_e^{new}$ are represented in the same semantic space and can be interpreted by the integration algorithm. For example, we can calculate the similarity or distance between $\vec{d}_c^{new}$ and $\vec{d}_e^{new}$ after the translation.

6.4. Experiment Setting

The translated category trees are then integrated by using the algorithm proposed in Chapter 4 for mono-lingual category tree integration. Most of the experiment settings are the same as we stated in Chapter 5. We use the same classifiers, the same evaluation methods and the same parameter values.

For the datasets we used in the experiment of cross-lingual category tree integration, please refer to Table 8. We collected 557 documents pairs in Chinese and English from the website of Hong Kong SAR. These documents cover a rather wide range of topics. All the documents pairs are then manually classified into 42 categories. These categories form two three-level category trees, one in Chinese and the other one in English respectively. The two category trees in different languages are the base category trees of the ten datasets and they have exactly the same structure. Then 10 datasets are derived from the two base category trees by selecting, splitting, combing different categories to form datasets. The Chinese and the English category tree of the same dataset have different structure for integration. And the structures of category trees between different dataset are totally different too. These structures cover most of the structures that can be used in our daily category trees and cover all the structures that our algorithm should deal with. The Chinese and the English category tree of the same dataset root at the same category too, as in the mono-lingual experiment. They will play as the role of source and master category tree respectively at first, and then vise verse.

Table 8. The cross-lingual datasets

	English Category Tree		Chinese Category Tree	
Data Source	Number of files	Number of Categories	Number of Files	Number of Categories
Dataset 1	196	14	244	13
Dataset 2	266	17	189	12
Dataset 3	287	21	194	11
Dataset 4	202	14	227	13
Dataset 5	213	14	220	13
Dataset 6	222	14	242	13
Dataset 7	203	14	206	13
Dataset 8	208	14	224	13
Dataset 9	213	14	210	13
Dataset 10	244	14	234	13
Average	225.4	15	219	12.7

6.5. Experiment Results

Table 9 shows the integration accuracy and some other statistical data of the experiment. "C->E" means the Chinese category tree acts as the source category and the English Category tree acts as the master category tree. "E->C" means the English category tree acts as the source category and the Chinese Category tree acts as the master category tree. The number of categories inserted takes up the most proportion of all the categories again as in the monolingual integration experiment. All the categories are assigned at lease one documents, so no empty category exists this time. The average category integration from Chinese to English is 83.50% and from English to Chinese is 82.63%. The average document integration accuracy from Chinese to English is 85.37% and from English to Chinese is 84.43%.

Table 9. Cross-lingual category tree integration result I

	$Accuracy_{category}$		$Accuracy_{doc}$		Number of Categories Split		Number of Categories Mapped		Number of Categories Inserted	
	C->E	E->C	C->E	E->C	C->E	E->C	C->E	E->C	C->E	E->C
Dataset 1	76.92%	78.57%	71.72%	80.69%	4	4	4	5	5	5
Dataset 2	91.67%	88.24%	96.83%	90.47%	4	6	4	6	4	5
Dataset 3	81.82%	80.95%	85.05%	81.34%	5	5	3	7	3	9
Dataset 4	76.92%	78.57%	80.62%	77.65%	2	2	4	5	7	7
Dataset 5	76.92%	71.43%	75.91%	79.43%	3	3	4	3	6	8
Dataset 6	92.31%	85.71%	90.91%	89.05%	3	4	4	4	6	6
Dataset 7	76.92%	78.57%	86.89%	81.45%	3	3	4	5	6	6
Dataset 8	92.31%	92.86%	95.54%	93.67%	2	4	5	2	6	8
Dataset 9	92.31%	92.86%	92.86%	89.20%	3	4	4	4	6	6
Dataset 10	76.92%	78.57%	77.35%	81.34%	3	4	2	4	8	6
Average	83.50%	82.63%	85.37%	84.43%	3.2	3.9	3.8	4.5	5.7	6.6

The recall and precision of the three operators are shown in Table 10 and Table 11. In Table 10 Chinese category tree acts as the source category tree and in and in Table 11 English category tree acts the source category tree. The F_1 values of *Split*, *Map* and *Insert* are 77.79%, 85.22% and 85.71% respectively in Table 10 and are 81.46%, 87.41% and 78.33% respectively in Table 11. The performance is quite good.

Table 10. Cross-lingual category tree integration result II

	Split			*Map*			*Insert*		
	Precision	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**
Dataset 1	50.00%	66.67%	57.14%	100.0%	66.67%	80.00%	75.00%	100.0%	85.71%
Dataset 2	75.00%	100.0%	85.71%	100.0%	100.0%	100.0%	100.0%	80.00%	88.89%
Dataset 3	60.00%	100.0%	75.00%	100.0%	60.00%	75.00%	100.0%	100.0%	100.0%
Dataset 4	100.0%	66.67%	80.00%	100.0%	80.00%	88.89%	57.14%	80%	66.66%
Dataset 5	66.67%	66.67%	66.67%	75.00%	75.00%	75.00%	83.33%	83.33%	83.33%
Dataset 6	100.0%	100.0%	100.0%	100.0%	80.00%	88.89%	83.33%	100.0%	90.91%
Dataset 7	66.67%	66.67%	66.67%	100.0%	80.00%	88.89%	66.67%	83.33%	74.07%
Dataset 8	100.0%	66.67%	80.00%	100.0%	100.0%	100.0%	83.33%	100.0%	90.91%
Dataset 9	100.0%	100.0%	100.0%	100.0%	80.00%	88.89%	83.33%	100.0%	90.91%
Dataset 10	66.67%	66.67%	66.67%	100.0%	50.00%	66.67%	75.00%	100.0%	85.71%
Average	78.50%	80.00%	77.79%	97.50%	77.17%	85.22%	80.71%	92.66%	85.71%

Table 11. Cross-lingual category tree integration result III

	Split			*Map*			*Insert*		
	Precision	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**
Dataset 1	75.00%	75.00%	75.00%	100.0%	100.0%	100.0%	60.00%	60.00%	60.00%
Dataset 2	83.33%	100.0%	90.91%	83.33%	83.33%	83.33%	100.0%	83.33%	90.91%
Dataset 3	80.00%	100.0%	88.89%	100.0%	77.78%	87.50%	66.67%	75.00%	70.59%
Dataset 4	100.0%	66.67%	80.00%	80.00%	80.00%	80.00%	71.43%	83.33%	76.92%
Dataset 5	66.67%	66.67%	66.67%	100.0%	75.00%	85.71%	62.50%	71.43%	66.67%
Dataset 6	75.00%	75.00%	75.00%	100.0%	80.00%	88.89%	83.33%	100.0%	90.91%
Dataset 7	66.67%	66.67%	66.67%	100.0%	83.33%	90.91%	66.67%	80.00%	72.73%
Dataset 8	75.00%	100.0%	85.71%	100.0%	66.67%	80.00%	100.0%	100.0%	100.0%
Dataset 9	75.00%	100.0%	85.71%	100.0%	80.00%	88.89%	100.0%	100.0%	100.0%
Dataset 10	100.0%	100.0%	100.0%	100.0%	80.00%	88.89%	50.00%	60.00%	54.55%
Average	79.67%	85.00%	81.46%	96.33%	80.61%	87.41%	76.06%	81.31%	78.33%

Both the values of integration accuracy and F_1 of the three operators are similar to that in the experiment of the mono-lingual integration. It proves that the cross-lingual concept space and thesaurus we constructed from the parallel corpus successfully help us to solve the semantic interoperability problem. Vectors of documents written in different languages can be projected to the same semantic space and further processed by the integration algorithm in the same way. After the language barrier between category trees in different languages is conquered, they can be integrated together as the mono-lingual category tree integration.

From Table 12 we can see the performance of the second step of *Split* operator in the cross-lingual integration. It gets an average F_1 value of 79.90% and 81.92% respectively. The value is very similar to s80.11% in the mono-lingual integration. This number is also acceptable. We can still see significant changes between different values as in the mono-lingual integration.

Table 12. Cross-lingual category tree integration result IV

	Precision		*Recall*		F_1	
	C->E	**E->C**	**C->E**	**E->C**	**C->E**	**E->C**
Dataset 1	100.0%	83.33%	66.00%	71.43%	79.52%	76.92%
Dataset 2	88.89%	100.00%	72.73%	83.33%	80.00%	90.91%
Dataset 3	80.00%	85.71%	66.00%	75.00%	72.33%	80.00%
Dataset 4	85.71%	80.00%	75.00%	66.67%	80.00%	72.73%
Dataset 5	100.0%	83.33%	71.43%	62.50%	83.33%	71.43%
Dataset 6	87.50%	100.00%	63.64%	85.71%	73.69%	92.31%
Dataset 7	85.72%	87.50%	75.00%	77.78%	80.00%	82.35%
Dataset 8	100.0%	83.33%	75.00%	83.33%	85.71%	83.33%
Dataset 9	83.33%	83.33%	71.43%	71.43%	76.92%	76.92%
Dataset 10	100.0%	100.00%	77.78%	85.71%	87.50%	92.31%
Average	91.12%	88.65%	71.40%	76.29%	79.90%	81.92%

Chapter 7. Conclusion and Future Work

Category tree integration is so important on internet and semantic web that more and more techniques are developed to resolve this problem. In this paper, we explore how to integrate mono-lingual and cross-lingual category trees by making use of implicit information embedded in the hierarch category tree structure.

We first survey many related work in data integration and cross-lingual related problems, such as ontology integration, schema matching, cross-lingual text classification and cross-lingual information retrieval. We formally defined category tree, category trees integration, cross-lingual category tree integration and three operators for integration. We correctly identify five category relationships between categories in the source and master category trees, namely, *Match*, *Disjoint*, *SubConcept*, *SuperConcept* and *Overlap*. Based on the category relationships, we develop six integration rules and propose a top-down level-based integration algorithm. For cross-lingual category integration, we construct a cross-lingual concept space from a parallel corpus by using associate constrain network. The experiments are conducted by using real web data from Yahoo, ODP and HKSAR websites. The results of both mono-lingual integration and cross-lingual integration show that our technique is promising. Contrary to the traditional ontology integration or schema matching, our work makes full use of the structure information contained in the category tree and the master category can learn from the source category tree to adjust its structure slightly.

Category tree integration is a novel and practical research topic in data integration. In our future work, we will look into more complicated structure, such as directed acyclic graph. Because of the complexity of the organization of information, categories in the directory or taxonomy may be duplicated many times or be relocated

to other positions. For example, a category usually has many parents instead of one and it can be the sibling of its parent if it is a very important category or it is big enough. Both Yahoo and ODP directory are not strict trees. More rules and improved algorithm are needed to solve these problems. For cross-lingual category tree integration, the data we collected for this experiment is not good enough. Some real tree structure should be used and more documents should be collected for further experiment. We should also further test the cross-lingual integration of source and master category tree with few document pairs.

Reference

[1] M. Abusalah, J. Tait and M. Oakes. Literature Review of Cross Language Information Retrieval. Transactions on Engineering, Computing and Technology, V4, February, 2005

[2] R. Agrawal and R. Srikant. On Integrating Catalogs. Proceedings of WWW10 Conference, May 1-5, 2001, Hong Kong, pp 603-612

[3] C. Alvarado, J. Teevan, M.S. Ackerman and D. Karger. Surviving the Information Explosion: How People Find Their Electronic Information. Technical Report AIM-2003-006, MIT AI Lab, 2003

[4] C. Apte, F.J. Damerau, and S.M. Weiss, Automated Learning of Decision Rules for Text Categorization, ACM Transaction on Information Systems, vol. 12, pp. 719-736, 1994

[5] P. Avesani, F. Giunchiglia, M. Yatskevich: A Large Scale Taxonomy Mapping Evaluation, In Proceedings of ISWC, 2005.

[6] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In Proc. conference on advanced information system engineering (CAiSE), 2002.

[7] N. Bel, C.H.A. Koster and M. Villegas. Cross-lingual text categorization. Proceedings of 7th European Conference on Research and Advanced Technology for Digital Libriies (ECDL'03), Trondheim, Norway, August, 126-139, 2003

[8] T.H. Cheng and C. Wei. Integration of Document-category Hierarchies: A Clustering-based Approach. Web 2003(The Second Workshop on e-Business), December 13-14, 2003, Seattle, Washington, USA

[9] N. Choi, I. Song and H. Han. A Survey on Ontology Mapping. SIGMOD Record, Wol. 35, No. 3, Sep. 2006

[10] W.W. Cohen and H. Hirsh, Joins that Generalize: Text Classification Using WHIRL, Proceedings of 4th International Conference on Knowledge Discovery and Data Mining (KDD-98), New York, NY, USA, 1998

[11] D. Davidov, E. Gabrilovich, S. Markovitch. Parameterized Generation of Labeled Datasets for Text Categorization Based on a Hierarchical Directory. In SIGIR'04, July 25-29, 2004, Sheffield, South Yorkshire, UK.

[12] H. Do, S. Melnik, E. Rahm: Comparison of Schema matching Evaluations In Proceedings of the GI-Workshop Web and Databases, Erfurt, 2002.

[13] A. Doan and A. Halevy: Semantic Integration Research in the Database Community: A Brief Survey AI Magazine, Special Issue on Semantic Integration, 2005.

[14] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos and A. Halevy. Learning to match ontologies on the Semantic Web. The VLDB Journal(2003)12:303-319

[15] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. Semantic Schema Matching. Technical Report DIT-05-014, University of Trento, 2005.

[16] D.A. Hull, Improving Text Retrieval for the routing problem using latent semantic indexing. In procedding of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval (Dublin, Ireland, 1994) 282-288

[17] R. Ichise, H. Takeda, and S. Honiden. Integrating multiple internet directories by instance-based learning. In IJCAI, pages 22-30, 2003

[18] T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Scholkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[19] Y. Kalfoglou, M. Schorlemmer: Ontology Mapping: The State of the Art The Knowledge Engineering Review Journal, 2003.

[20] D. Kim, J. Kim and S. Lee. Catalog Integration for Electronic Commerce through Category-Hierarchical Merging Technique. Proceedings of the 12th Int'l Wrkshp

on Research Issues in Data Engineering: Engineering e-Commerce/e-BUsiness Systems(RIDE'02)

[21] H.J. Kim and S.G. Lee, A semi-supervised Document Clustering Techniques for Information Organization, Proceedings of the ACM 9th International Conference on Information and Knowledge Management, McLean, VA, 2000

[22] M.S. Lacher and G. Groh, Facilitating the exchange of explicit knowledge through ontology mappings. Proceedings of the 14th FLAIRS Conference. AAAI Press, 2001

[23] H. Li and K. Yamanishi, Text Classification using ESC-Based Stochastic Decision Lists, Proceedings of the 8th ACM International Conference on Information and Knowledge Management, Kansac City, Mo, UAS, 1999.

[24] K. W. Li and C. C. Yang. Conceptual Analysis of Parallel Corpus Collected from the Web. Journal of the American Society for Information Science and Technology, vol.57, no.5, 2006, pp.684-696.

[25] K. W. Li and C. C. Yang. Automatic Cross-lingual Thesaurus Generated from the Hong Kong SAR Police Department Web Corpus for Crime Analysis. Journal of the American Society for Information Science and Technology, vol.56, no.3, February, 2005, pp.272-282.

[26] J. Madhavan, P.A. Bernstein, E. Rahm, Generic Schema Matching with Cupid, Proceeding of 27th International Conference On Very Large Database, pp.49-58, 2001

[27] D.L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning, pages 483--493, San Francisco, USA, 2000. Morgan Kaufmann.

[28] N.F. Noy: Semantic Integration: A Survey of Ontology-based Approaches Sigmod Record, Special Issue on Semantic Integration, 2004.

[29] N.F. Noy and M.A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In Proceedings of the National Conference on Artificial Intelligence (AAAI), 2000.

[30] N.F. Noy and M.A. Musen. Anchor-PROMPT: Using Non-Local Context for Semantic Matching. In Proceedings of workshop on OIS at IJCAI, 2001

[31] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. The VLDB Journal 10(4):334-350, 2001

[32] I. Ryutaro, T. Hideaki, H. and H. Shinichi. Rule Induction for Concept Hierarchy Alignment. in Proceedings of the Workshop on Ontologies and Information Sharing at the 17th International Joint Conference on Artificial Intelligence (IJCAI), Seattle, WA, 2001, 26-29

[33] F. Sebastiani. Machine Learning in Automated Text Categorization. In ACM Computing Surveys, Vol. 34, No. 1, 1-47, 2002.

[34] P. Shvaiko1 and J. Euzenat2, A Survey of Schema-based Matching Approaches. Journal on Data Semantics, 4:146–171, 2005

[35] A. Sun, E.P. Lim and W.K. Ng. Performance Measurement Framework for Hierarchical Text Classification. In Journal of the American Society for Information Science and Technology (JASIST), 54(11), 1014-1028.

[36] X. Su and J.A. Gulla. An Information Retrieval Approach to Ontology Mapping. Data & Knowledge Engineering, 58 (2006) 47-69

[37] C. Wei and T.H. Cheng. A Clustering-Based Approach for Supporting Document-Category Integration, Proceedings of the the 7th Pacific Asia Conference on Information Systems (PACIS'03), Australia, 2003

[38] C. C. Yang and K. W. Li. A Heuristic Method Based on a Statistical Approach for Chinese Text Segmentation. Journal of the American Society for Information Science and Technology, vol.56, no.13, November, 2005, pp.1438-1447.

[39] C. C. Yang and K. W. Li. Automatic Construction of English/Chinese Parallel Corpora. Journal of the American Society for Information Science and Technology, vol.54, no.8, June, 2003, pp.730-742

[40] C. C. Yang and K.W. Li. Cross-lingual information retrieval: The Challenge in multilingual digital libraries, Design and Usability of Digital Libraries: Case Studies in the Asia Pacific, Y.L. Theng and S. Foo (eds.), Idea Group, Inc. 2004

[41] C. C. Yang and K. W. Li, An Associate Constraint Network Approach to Extract Multi-lingual Information for Crime Analysis. Decision Support Systems, accepted for publication.

[42] C. C. Yang, C. Wei, and K. W. Li, "Cross-lingual Thesaurus for Multilingual Knowledge Management," Decision Support Systems, accepted for publication.

[43] C. C. Yang and J. Lin. Integrating Web Directories by Learning their Structures. Proceedings of the International World Wide Web Conference (WWW'07), Banff, Alberta, Canada, May 8-12, 2007.

[44] C. C. Yang and J. Luk, Automatic Generation of English/Chinese Thesaurus Based on a Parallel Corpus in Law. Journal of the American Society for Information Science and Technology, Special Topic Issue on Web Retrieval and Mining: A Machine Learning Perspective, vol.54, no.7, May, 2003, pp.671-682.

[45] C. C. Yang, C. Wei, and K. W. Li. Cross-lingual Thesaurus for Multilingual Knowledge Management. Decision Support Systems, accepted for publication.

[46] D. Zhang and W.S. Lee. Web Taxonomy Integration using Support Vector Machines. WWW 2004, May 17-22, 2004, New York, USA

[47] S. Zhu, C.C. Yang and W. Lam. CatRelate: A New Hierarchical Document Category Integration Algorithm by Learning Category Relationship. ICADL 2004, LNCS 3334, pp. 280-289, 2004