TECTAS: Bridging the Gap Between Collaborative Tagging Systems and Structured Data

by

Seyyed Ali Moosavi

B.Sc. in Computer Engineering, Sharif University of Technology, 2008

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

October 2010

Abstract

Ontologies are core building block of the emerging semantic web, and taxonomies which contain class-subclass relationships between concepts are a key component of ontologies. A taxonomy that relates the tags in a collaborative tagging system, makes the collaborative tagging system's underlying structure easier to understand. Automatic construction of taxonomies from various data sources such as text data and collaborative tagging systems has been an interesting topic in the field of data mining.

This thesis introduces a new algorithm for building a taxonomy of keywords from tags in collaborative tagging systems. This algorithm is also capable of detecting has-a relationships between tags. Proposed method — the TECTAS algorithm — uses association rule mining to detect is-a relationships between tags and can be used in an automatic or semi-automatic framework. TECTAS algorithm is based on the hypothesis that users tend to assign both "child" and "parent" tags to a resource. Proposed method leverages association rule mining algorithms, bi-gram pruning using search engines, discovering relationships when pairs of tags have a common child, and lexico-syntactic patterns to detect meronyms.

In addition to proposing the TECTAS algorithm, several experiments are reported using four real data sets — Del.icio.us, LibraryThing, CiteULike, and IMDb. Based on these experiments, the following topics are addressed in this thesis: (1) Verify the necessity of building domain specific taxonomies (2) Analyze tagging behavior of users in collaborative tagging systems (3) Verify the effectiveness of our algorithm compared to previous approaches (4) Use of additional quality and richness metrics for evaluation of automatically extracted taxonomies.

Preface

This thesis is a result of a collaborative research between four people: Seyyed Ali Moosavi, Laks V.S. Lakshmanan, Rachel Pottinger, and Tianyu Li. Plural pronouns are used when referring to the author(s) through the thesis to emphasis team work. Details of contribution made by each co-author is as follows:

Dr. Rachel Pottinger and Dr. Laks V.S. Lakshmanan were my supervisors through my master's program. The design of the research was done by my supervisors. My supervisors helped me both in finding and choosing relevant references, and in developing novel ideas through weekly meetings. They also helped me in choosing valuable experiments. Both Dr. Pottinger and Dr. Lakshmanan helped in writing up and formatting the thesis.

Tianyu Li later added the idea of adding relationships when pairs of tags have a common child, and also the idea of using the hitrate metric in the experiments section. She was also involved in writing up the aforementioned sections in the thesis.

I finalized the idea of using association rule mining in TECTAS algorithm, implementing the algorithm, and optimizing it. I was in charge of performing surveys to make sure all related work are cited properly. I also implemented four other competing algorithms. I ran the experiments and I was involved in writing up the thesis. Finally, I was involved in analysis of the output of algorithms together with Tianyu Li.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I am heartily thankful to my supervisors Dr. Rachel Pottinger and Dr. Laks Lakshmanan, whose encouragement, guidance and support from the initial to the final level, made this work possible.

I also thank George Tsiknis for reading this thesis and for his insightful suggestions. I would also like to thank the members of the data management and mining lab, for all their help with my questions and for providing a friendly research environment in the lab.

Ali Moosavi

Dedication

To my beloved father and mother; for their understanding and endless love through the duration of my studies.

Chapter 1

Introduction

1.1 Collaborative Social Tagging Systems

In this chapter collaborative tagging systems (CTSs) are introduced in Section 1.1.1 with some examples of CTSs from web. Formal definition of CTSs is introduced in Section 1.1.2. Section 1.2 is an introduction to ontologies and taxonomies; also WordNet as the most popular ontology is introduced in this section. Association rule mining algorithms are introduced briefly in Section 1.3. For a detailed discussion on association rule mining algorithms please refer to Chapter 2. Finally, the contributions of this thesis are mentioned in Section 1.5.

1.1.1 What is a Collaborative Social Tagging System?

Collaborative social tagging systems such as Del.icio.us (for tagging bookmarks), YouTube (for tagging videos), Flickr (for tagging photos), IMDb (for tagging movies), LibraryThing (for tagging books) and CiteULike (for tagging publications), have become popular in recent years. These systems permit users to bookmark and tag resources (documents, photos, videos, etc.) and share them with other people and friends.

The detailed information about some CTSs are as follows:

DEL.ICIO.US: Del.icio.us is a social bookmarking web service for storing, sharing, and searching web bookmarks. Because they are stored centrally, Del.icio.us makes it easy to synchronize bookmarks between different computers. Users of Del.icio.us are growing rapidly, e.g. tagging actions of users in Del.icio.us have grown more than 1000 times from September 2003 to December 2007.

IMDb: IMDb, the Internet Movie Database, is a collection of movie information which catalogs details about movies, e.g. cast, director, writer, genre, and keywords assigned by users. IMDb users can enter arbitrary keywords(tags) for the movies, but the information about the taggers is not stored. From a collaborative tagging system view, IMDb dataset is a collection of movies and their associated tags.

LibraryThing: LibraryThing is a social cataloging web application used for storing and sharing library catalogs and book lists. LibraryThing was initially started in August 2005 and in September 2009 the web site had 920,000 users and had catalogued more than 45 million books. Users of LibraryThing can import information from more than 680 libraries, including the Canadian National Catalogue and the Library of Congress [3].

CiteULike: CiteULike is a web site that offers a free service for storing, organizing, and sharing scholarly papers. CiteULike allows users to store papers with their metadata, abstracts, and links to the papers at the publishers' web sites. Users can also provide personal comments and tags about the papers. CiteULike does not provide access to the full text of articles.

Aforementioned CTSs cover a distinct range of resources and because of the diversity of these four CTSs, we have used these CTSs in the experiments section of the thesis.

Given the richness of tagging data, researchers have studied numerous problems on tagged data, including taxonomy creation, search, and tag recommendation. Discovering taxonomies from CTSs is interesting because the extracted taxonomy can be deployed on the very CTS that it came from, and in turn make it easier for users to discover content and search through resources. This is particularly important in the case of CTSs containing non-textual resources like photos and videos.

1.1.2 Formal Definition of CTS

A *collaborative tagging system* [36] is a 4-tuple $C = (U, T, I, Y)$ where U is a set of users, T is the set of tags used by the users, I is the set of items (resources) to which tags are assigned, and Y, the set of tag assignments, is a ternary relation on tags, users, and items, i.e., $Y \subseteq U \times T \times I$.

Specific instances of CTSs may slightly vary from our definition above. E.g. in IMDb, user information is not available. We can model IMDb by dropping U and defining $Y \subseteq T \times I$ as a binary relation. In some CTSs such as bibsonomy [24], users can declare their own is-a relationships between keywords, but this is not the norm, and is not a well used feature.

1.2 Ontologies and Taxonomies

An ontology is a representation of a set of concepts in a domain and the relationships between those concepts. From the time ontologies emerged they have shown their usefulness in different application scenarios such as the semantic web, artificial intelligence, software engineering, and biomedical informatics.

An ontology is an explicit, formal specification of a shared conceptualization [17]. Being formal implies that the ontology should be machine readable and being shared means ontology must be accepted by people in a community. As mentioned in [5] components of ontologies include:

- Individuals (instances, objects): An individual can be any concrete or abstract object in the world.
- Classes (sets, concepts): Classes are set of individuals.
- Attributes (properties, features): Attributes can be assigned to classes or individuals.
- Relations: The ways classes and individuals are related to each other.
- Rules: If-then statements that describe logical inferences.

Other components of ontologies include function terms, restrictions, axioms and events. In this thesis only the classes and relations between classes (mainly as is-a and has-a relationships) are discussed.

Several formal languages are introduced to encode ontologies and knowledge about specific domains, such as: DOGMA (Developing Ontology Grounded Methods and Applications), KIF (Knowledge Interchange Format), OKBC (Open Knowledge Base Connectivity), and OWL (Web Ontology Language) [4]. Among these ontology languages, OWL is the most commonly used ontology language. OWL ontologies are serialized using Resource Description Framework (RDF) syntax. Several languages are emerged for querying RDF graphs; the most common language is SPARQL which is a SQL like language and is recommended by W3C.

Ontologies organize information in content management systems and are building blocks of the emerging Semantic Web. Taxonomies — hierarchical classifications of concepts via class-subclass relationships — are a key component of ontologies. Given the importance of taxonomies, substantial work has been done in extracting taxonomies automatically from large repositories like text corpora, databases, and the web to complement and enrich human-generated ontologies.

IR (Information Retrieval) techniques and keyword queries have been successfully used in web search and document search. Despite the enormous success of IR techniques in document and web search, discovering content in a CTS by simple keyword search is severely limited in its scope since the resources typically possess a small set of tags, which tend to be sparse. In other words, contents that are tagged are either non textual or have very limited keywords associated to them (such as bookmark titles). On the other hand, a taxonomy can help make sense of the tags in a CTS and search for relevant information. For example, if the tags in a CTS have been arranged in a taxonomy, browsing the taxonomy can help the users better refine their queries, either to find more related items by using a more general term or to find fewer related items by using a more specific term.

In principle, we could use a general purpose taxonomy such as WordNet[1] to browse as CTS. There are two disadvantages with this. First, tags in CTSs are not based on a fixed vocabulary but constantly evolve. Thus, one cannot expect WordNet (or similar systems) to capture the vocabulary in a dynamic CTS. For example, many computer related tags (e.g., "Mac OS X") are not found in general purpose taxonomies like WordNet. Secondly, and somewhat surprisingly, as we

[1]http://wordnet.princeton.edu; WordNet is richer than a taxonomy: it also includes hyponyms, synsets, etc.

demonstrate in Section 5.2, even when terms corresponding to tags in a CTS *are* present in WordNet, in many cases, valid is-a-relationships between them are missing in WordNet. This mirrors a similar finding in the case of taxonomy extracted from Wikipedia using YAGO [39], where it was found that using a combination of WordNet and Wikipedia found significantly more ontological relationships (including is-a) that were not present in WordNet.

1.2.1 WordNet

As defined by [1], WordNet is a lexical English database which provides a variety of semantic relations defined between concepts. WordNet groups words into synsets — sets of synonym words — and stores the relations between synsets. Using WordNet, information can be derived about relations between nouns, verbs and adjectives. As an example, information about noun relations include:

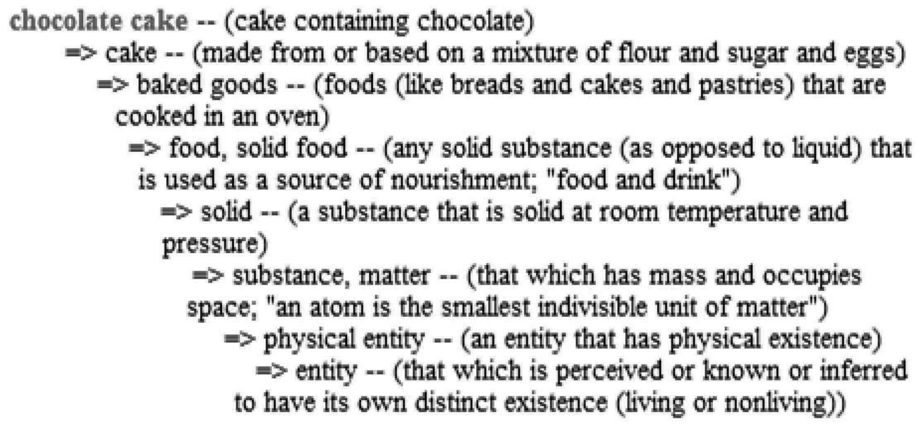
- Synonymy: similar meanings.
- Antonymy: opposite meanings.
- Hyponymy: X is-a-kind-of Y, where X is a more specific concept.
- Meronymy: X is-a-part-of Y, where X is a concept that represents a part of concept Y.

Figure 1.1 depicts a part of hierarchical information derived from WordNet. In this figure, the word "chocolate cake" is subclass of "cake" and "cake" itself is subclass of "baked goods".

1.3 Association Rule Mining Algorithms

Association rule mining algorithms (ARM) are used to extract *frequent itemsets* and *interesting rules* from a set of transactions along with associated items for each transaction. In this section we introduce these concepts using the typical example in the literature: Market basket analysis [18].

Market basket analysis is the process of analyzing customers buying habits which is done by discovering their tendency to buy items together. Merchants can benefit from these associations between items in their marketing strategy. Several



```
chocolate cake -- (cake containing chocolate)
    => cake -- (made from or based on a mixture of flour and sugar and eggs)
       => baked goods -- (foods (like breads and cakes and pastries) that are
        cooked in an oven)
          => food, solid food -- (any solid substance (as opposed to liquid) that
           is used as a source of nourishment; "food and drink")
             => solid -- (a substance that is solid at room temperature and
             pressure)
                => substance, matter -- (that which has mass and occupies
                space; "an atom is the smallest indivisible unit of matter")
                   => physical entity -- (an entity that has physical existence)
                      => entity -- (that which is perceived or known or inferred
                     to have its own distinct existence (living or nonliving))
```

Figure 1.1: An example of noun hierarchies from WordNet

strategies can be used such as (1) putting these items in a package to make shopping easier for customers (2) putting these items in the furthest distance in the store, so customers are likely to check the aisles in between and buy more items (3) avoid putting items in the same basket on promotion since it will not probably create increase in profit [2, 18].

As an example, if customers interested in buying milk are also interested in buying bread, this can be shown as the following association rule:

$$\text{milk} \quad \Rightarrow \quad \text{bread} \qquad [\text{support} = 10\%, \text{confidence} = 80\%]$$

Interesting rules are identified by two measures: support and confidence. Referring to example above, support for this rule reflects what percentage of all the transactions are transactions that contain both milk and bread (10%). Confidence for this rule means what percentage of customers who bought milk, also bought bread (80%). Association rules that have support greater than minimum support threshold and confidence greater than minimum confidence threshold are called interesting (strong) rules. Thresholds for support and confidence are usually set by users and are domain specific thresholds [18].

1.3.1 Formal Definition of Association Rule Mining

Let $I = \{I_1, I_2, \cdots, I_n\}$ be the set of all items in the universe of our problem, and let D be the set of transactions where each transaction T is a set of items (i.e. $T \subseteq I$). An association rule is an implication of the form $X \Rightarrow Y$, with following conditions:

$$X \subseteq I,\ Y \subseteq I,\ X \cap Y = \emptyset$$

Itemset refers to a set of items from set I. Absolute support (frequency) of an itemset is the number of transactions that contain that itemset. Relative support (or simply support) of an itemset X is defined as the proportion of transactions containing itemset X. If relative support of an itemset X is greater than the minimum support threshold, then itemset X is a frequent itemset [18].

Support of a rule is defined as percentage of transactions in D that contain both A and B, i.e.:

$$\text{Support}(X \Rightarrow Y) = P(X \cup Y)$$

Confidence of a rule is defined as the percentage of transactions in D containing A, that also contain B:

$$\text{Confidence}(X \Rightarrow Y) = P(Y \mid X) = \frac{\text{Absolute\_Support}(X \cup Y)}{\text{Absolute\_Support}(X)}$$

Several algorithms have been developed for extracting frequent itemsets such as: Apriori [6], FP-growth [19], Eclat [43] and OPUS search [40]. Among these algorithms, the Apriori and the FP-growth algorithm are the most popular, and are the most efficient algorithms in many data domains. In this thesis we use FP-growth algorithm which is faster than the basic Apriori algorithm in our proposed architecture [18].

Association rule mining algorithms first find all of the frequent itemsets which are itemsets with support bigger than minimum support threshold. In this step algorithms such as Apriori and FP-growth can be used. Then, interesting rules are generated from those itemsets. The result is the rules with support and confidence bigger than predefined thresholds. These two steps are described in detail in Chapter 2.

1.4 The Problem

This thesis studies the following problem: given a collaborative tagging system consisting of users, resources (also called items), and tags assigned by users to items, efficiently extract a taxonomy over the tags in the CTS. The extracted taxonomy should be rich and have a high quality — both terms are defined precisely later.

The problem of building taxonomy for collaborative tagging systems is formalized as follows:

Input: A collaborative tagging system.

Optional Input: User's defined is-a relationships.

Output: A taxonomy $T(N,E)$ of keywords represented as a DAG. N is the set of nodes representing concepts. E is the set of edges representing is-a relationships between concepts where $e(g,s) \in E$ is an edge where g is the generalization in the is-a relationship and s is the specialization.

As mentioned in Section 1.1.2, Some collaborative tagging systems e.g. bibsonomy also allow users to provide a set of is-a relationships between tags. The TECTAS algorithm proposed in this thesis will use these relationships if provided. However, we do not require them since most sources do not have them. Additionally, from over 10,000 is-a relationships entered by users of bibsonomy, most of them were spam; hence an algorithm can not use these tags unless a human validates these tags beforehand.

Based on Section 1.1.2, CTSs are usually in form of 4-tuple $C = (U,T,I,Y)$; however, there exist collaborative tagging systems in form of a 3-tuple $C = (T,I,Y)$, i.e. $U = \emptyset$. Based on the capability to build the taxonomy from 3-tuple CTSs, a suitable taxonomy constructing algorithm should be selected. The TEC-

TAS algorithm proposed in this thesis is capable of building the taxonomy in such cases.

This thesis studies how to efficiently extract is-a relationships between tags in a given CTS. General purpose taxonomies such as WordNet miss many relationships between words; therefore they cannot be used to check the correctness of the taxonomies built from collaborative tagging systems. The algorithm should be robust against the use of bi-grams (more generally, *n*-grams) as tags and should achieve a high precision. An absolute notion of recall is hard to measure for large CTSs having tens of thousands of tags since the set of true is-a relationships is not available beforehand and are too many to detect manually. Instead, in Chapter 5, we use a notion of relative recall between the various algorithms being compared. In addition to precision and relative recall which measure the accuracy of the taxonomy extracted, we also use measures for its richness: maximum depth, average number of children, and average depth of the tree/DAG taxonomy.

As an extension to our TECTAS algorithm, we have also developed a method to detect terms with has-a relationship between them. To our knowledge, extracting has-a relationships from CTS is a novel area; hence, we separate is-a detection and has-a detection algorithms and their evaluations. The reason is to focus more on the taxonomies and is-a relationships which are more interesting in this thesis.

1.5 Contributions

We make the following specific contributions in this thesis:

- We propose an algorithm for taxonomy extraction from a CTS, called TECTAS (for Taxonomy Extraction from Collaborative TAgging Systems).
- We thoroughly explore how to use association rule mining over CTS tags to extract a taxonomy and show that one form of association mining significantly outperforms the others.
- We describe a method of pruning invalid edges from the taxonomy based on discovering bi-grams.
- We describe a method for recovering valid pruned edges when two items in the taxonomy share a common child.

- We provide an extension to our TECTAS algorithm to discover terms with has-a relationship between them.
- We describe a prototype system based on TECTAS for taxonomy extraction.
- We conducted a comprehensive set of experiments on four real data sets comparing our algorithm with previous work with regarding to quality and richness of the extracted taxonomy. In the process we augmented the existing set of metrics by proposing new metrics of our own.
- In addition to showing the effectiveness of TECTAS compared to previous works, our experiments confirm the following: (i) general purpose taxonomies like WordNet may miss many valid is-a relationships, even when both tags involved in the relation are present; (ii) the hypothesis that users tend to use both parent and child tags (from a taxonomy in their mind) when tagging resources; (iii) more frequent terms usually correspond to more general terms in the taxonomy.

1.6 Thesis Structure

In this chapter, Ontologies and ARM algorithms were introduced as well as the problem statement and the contributions of this thesis (see Sections 1.4 and 1.5). Chapter 2 provides the required background needed for understanding the ARM phase of TECTAS algorithm. Chapter 3 discusses related work and introduces competing algorithms to TECTAS. Chapter 4 introduces our proposed taxonomy building architecture and describes the TECTAS algorithm's phases in detail. Chapter 5 shows our extensive experiments, including comparing our algorithm with previous methods on real datasets — Del.icio.us, IMDb, LibraryThing, and CiteU-Like. Finally, Chapter 6 concludes and discusses future work.

Chapter 2

Background

In this Chapter two main steps in mining association rules are discussed: (1) finding all frequent itemsets (2) finding interesting rules from frequent itemsets. In Section 2.1 the Apriori algorithm is discussed in detail and the differences between the Apriori algorithm and the FP-growth algorithm are mentioned. Section 2.3 discusses an example of the Apriori algorithm's phases.

2.1 Apriori Algorithm

In this section we introduce the Apriori algorithm. FP-growth algorithm is generally a more efficient version of Apriori algorithm which will be discussed at the end of this section. For a detailed discussion on Apriori algorithm please refer to [18], which was used as the primary reference for preparing this section of thesis.

Apriori algorithm is based on an antimonotone property of itemsets called the Apriori property: Any subset of frequent itemsets must be frequent. This property is called antimonotone because if a set fails in a test, all of its supersets fail in that test as well.

Proof of the Apriori property is simple: If an item I is not frequent (i.e. $P(I) < min\_support$), then for any set X, $P(I \cup X) < min\_support$. That is true because if an item is added to a set, the frequency of the resulting set will be less or equal to the original set.

The Apriori algorithm uses an iterative approach to build (k+1)- frequent itemsets from k-itemsets. First, frequent 1-itemsets (called L_1) are found by accumulating the count of each item in one database scan. Then, L_1 is used to find L_2 and so on, until no more frequent itemsets of size k can be found.

Each phase of building L_{m+1} from L_m consists of two steps: a join step and a prune step. In the join step, to find L_{m+1}, a set of candidate (m+1)-itemsets (called C_{m+1}) is generated by joining L_m with itself. Apriori assumes that itemsets are sorted in lexicographic order. Let l_1 and l_2 be two itemsets in L_m and assume $l_i[j]$ refers to the j-th item in l_i. When calculating $L_m \bowtie L_m$, one of the itemsets generated by l_1 and l_2 will be:

$$l_1[1], l_1[2], \cdots, l_1[m], l_2[m]$$

In the prune step, C_{m+1} which contains the potential frequent itemsets, is scanned to determine the count of each candidate in C_{m+1}. The result will be L_{m+1} in which all the itemsets have a support grater than minimum support. To speed up this step, a hash tree of all frequent itemsets is stored; therefore, size of C_{m+1} can be reduced by removing non-frequent m-itemsets from it with help of this hash tree. the Apriori algorithm is shown in Algorithm 2.1 which is a summarized version of the algorithm in [18].

Frequent-Pattern growth (FP-growth) algorithm is a modified version of the Apriori, in which frequent itemsets are extracted without generating candidates. Two issues exist in candidate generation phase of the Apriori algorithm which significantly affects its performance: 1) The number of candidates to be generated can become numerous, making the Apriori algorithm computationally impossible to be used. As an example, for a frequent pattern with size of 100, the total number of candidates to be generated is at least $2^{100} \approx 10^{30}$. 2) Repeated scans of the database to check each transaction and calculate support of candidate itemsets is computationally expensive [18].

As discussed in [18], the idea behind FP-growth method is to improve the Apriori algorithm by skipping candidate generation phase. In FP-growth method, database is compressed in to a prefix tree called FP-tree (Frequent-Pattern tree) which resides in memory instead of hard disk. Then the database is divided into a set of conditional databases. Each of these conditional databases is representing

Algorithm 2.1 Apriori Algorithm

```
Input: (D) A database of transactions
    (min_sup) Minimum support count threshold
Output: (L) frequent itemsets in D
 1: $L_1$ = frequent 1-itemsets
 2: for $m = 1$ to $\infty$ do
 3:     if $L_m = \emptyset$ then
 4:         break
 5:     end if
 6:     $C_{m+1}$ = candidates generated from $L_m$
 7:     for transaction T in database do
 8:         increment count of all candidates in $C_{m+1}$ that are contained in T
 9:         $L_{m+1}$ = candidates in $C_{m+1}$ with support $\geq$ min_support
10:     end for
11: end for
12: return L = $\bigcup_m L_m$
```

one frequent item or pattern fragment. Conditional database for an itemset *I* is all of the transactions that contain *I*, where elements of *I* are removed from those transactions. At this step, frequent mining on each database is done separately. For a detailed description of FP-growth algorithm and divide-and-conquer method used in this algorithm, please refer to [18].

2.2 Generating Rules from Frequent Itemsets

As mentioned in [18], after frequent itemsets are generated, association rules can be extracted using following procedure:

First, we generate all nonempty subsets of each frequent itemset f. Then for every nonempty subset f_i of f:

$$\text{if } \frac{support\_count(f)}{support\_count(f_i)} \geq min\_conf \text{ then } f_i \Rightarrow (f - f_i) \text{ is a strong rule.}$$

Since rules are generated from frequent itemsets, their support is automatically greater or equal than minimum support.

Table 2.1: Transactional data in database D

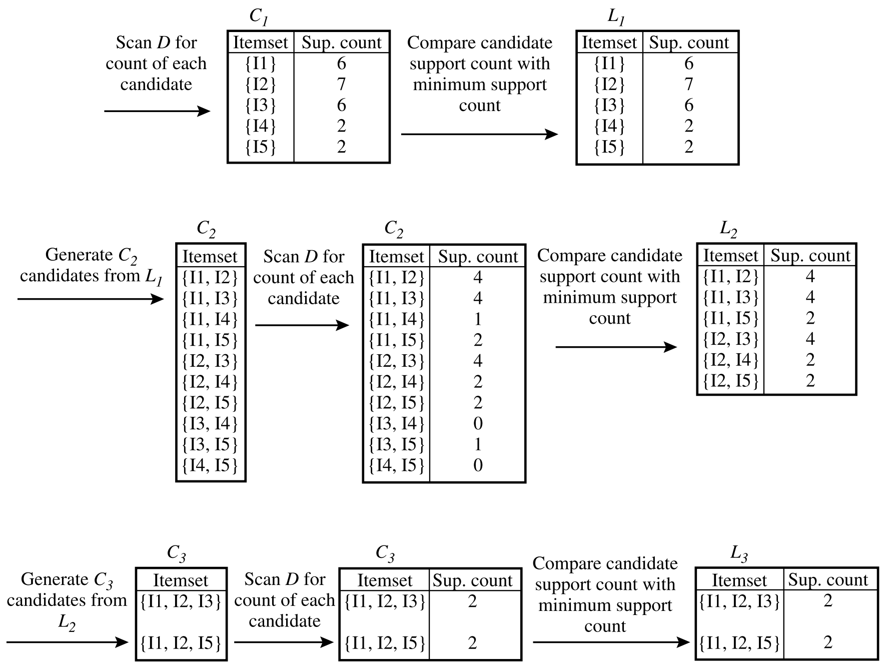
Transaction ID	Items
T1	I1, I2, I5
T2	I2, I4
T3	I2, I3
T4	I1, I2, I4
T5	I1, I3
T6	I2, I3
T7	I1, I3
T8	I1, I2, I3, I5
T9	I1, I2, I3

2.3 Example of Rule Mining Using Apriori

This example is a concise version of the example in [18]. Consider database D (table 2.1) consisting of 9 transactions. Suppose minimum support count is 2 (i.e. minimum support is 2/9 = 22%) and minimum confidence is 70%. To extract the interesting rules, first frequent itemsets are found using the Apriori algorithm. Next, association rules are generated using support and confidence thresholds.

Steps for generating frequent itemsets using Apriori are shown in Figure 2.1 and are as follows:

- Generating 1-itemsets: L_1 consists of the candidate 1-itemsets that occur more than two times in the database. As shown in figure 2.1, in this phase $L_1 = C_1$.
- Generating 2-itemsets: In this phase $L_1 \bowtie L_1$ is performed to generate C_2. Then, transactions in D are scanned and support count for each candidate itemset in C2 is accumulated. Finally L_2 is generated by picking 2-itemsets that have minimum support greater than the threshold.
- Generating 3-itemsets: In this phase $L_2 \bowtie L_2$ is performed to generate C_3. C_3 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}. Now, the Apriori property is used in the pruning phase for C_3.


Scan D for count of each candidate
C_1
Compare candidate support count with minimum support count
L_1
Generate C_2 candidates from L_1
C_2
Scan D for count of each candidate
C_2
Compare candidate support count with minimum support count
L_2
Generate C_3 candidates from L_2
C_3
Scan D for count of each candidate
C_3
Compare candidate support count with minimum support count
L_3

C_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

C_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Figure 2.1: Candidate itemset and frequent itemset generation in Apriori algorithm [18]

Using Apriori property we can determine that from 6 candidates in C_3 only 2 of them can be potential frequent itemsets. For example, for {I2, I3, I5}, 2-item subsets are {I2, I3}, {I2, I5} and {I3,I5}. However, {I3, I5} is not a member of L_2 and hence it is not frequent based on the Apriori Property. Therefore, C3= {{I1, I2, I3}, {I1, I2, I5}} after pruning step. Finally the transactions in D are scanned in order to determine L_3.

- Generating 4-itemsets: In this phase $L_3 \bowtie L_3$ is performed to generate C_4. The join results in {{I1, I2, I3, I5}}, but this itemset is pruned since its subset {{I2, I3, I5}}is not frequent. As a result, $C_4 = \emptyset$ and the algorithm terminates.

At this point, the Apriori algorithm is complete and all frequent itemsets are found. The next step is finding strong association rules that satisfy both minimum support and minimum confidence.

Using Apriori, 13 frequent itemsets were mined: L = {{I1}, {I2}, {I3}, {I4}, {I5}, {I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}. Lets consider {I1,I2,I5} which has 6 nonempty subsets: {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, and {I5}. Since the minimum confidence is 70%, accepting or rejecting strong rules are as follows:

I1, I2 $\Rightarrow$ I5 is rejected since the confidence for this rule is support{I1,I2,I5}/support{I1,I2} = 2/4 = 50% which is less than 70%.

With the same calculations, following rules are accepted:

- I2, I5 $\Rightarrow$ I1 (confidence = 100%)
- I1, I5 $\Rightarrow$ I2 (confidence = 100%)
- I5 $\Rightarrow$ I1, I2 (confidence = 100%)

and these rules are rejected:

- I2 $\Rightarrow$ I1, I5 (confidence = 29%)
- I1 $\Rightarrow$ I2, I5 (confidence = 33%)

The same procedure should be repeated for all members of L to extract all strong rules.

Chapter 3

Related Work

There are many definitions for ontologies. We define an ontology as a formal description of concepts in a domain and relationships between concepts such as is-a and part-of.

A number of works focus on extracting ontologies from the web such as [34, 41] where the authors learn non-taxonomic relationships (e.g., “cures”) between words. In [39], the authors complement general purpose ontologies such as WordNet by leveraging Wikipedia. Our work and interest in this thesis is complementary to these works as we focus on CTSs, which are fundamentally different. Also, our interest is extracting taxonomic relations from the input corpus without using general purpose ontologies.

Some recent works extract ontologies or taxonomies from CTSs. Some proposed approaches [8, 29, 31] match CTS tags to existing concepts in general purpose ontologies such as WordNet, resulting in a graph of tags. Because existing ontologies are made by experts, these methods have high accuracy; however, because CTSs have a dynamic ad-hoc usage of terms, general purpose ontologies miss many edges. As an example, our experiments show that WordNet is missing more than 25% of correct edges between the concepts extracted from Del.icio.us, even when both the parent and child concepts are represented in WordNet.

Schmitz [37] uses conditional probabilities between pairs of tags, the number of users using each tag, and the number of resources containing each tag to find tag pairs. Constructed pairs are then added to a weighted graph where edge weights represent how often paths from the leaves to the root go through that edge. For each leaf, the path to the root with the highest average weight is chosen. The final tree is built by integrating these paths. Schmitz's algorithm cannot identify the exact relationship (e.g., is-a, part-of, synonym, etc.) between terms; it simply determines if two terms are related by 'any' or not. By contrast, our algorithm pinpoints is-a relationships between terms.

A CTS can be naturally regarded as a tripartite graph between users, items, and tags. Mika [32] derives two bipartite graphs from the tripartite graphs of CTS; graph of items and tags, and graph of users and tags. Using set theoretic concepts, a classification hierarchy is built from the graph of items and tags, and a hierarchy based on sub-community relationships is built from the graph of users and items.

Heymann and Garcia-Molina [22] create a taxonomy tree by building a series of vectors, where each vector's dimension is the number of tagged URLs in the dataset. They use cosine similarity to find distance between tags, and then calculate centrality for each tag in the tag graph. Authors use betweenness centrality to find centrality of each tag in the graph. Betweenness centrality for vertex v can be calculated as follows: Divide the number of all possible shortest paths in the graph that pass through v by all possible shortest paths in the graph. Their algorithm organizes tags into a tree by starting with the *root* node, and adding other tags in decreasing order of centrality to the tree. Their method has the following limitations: (i) they find trees rather than DAGs; (ii) sparseness of tag vectors causes tags around the leaves to be far from the topic of the root; (iii) they assume all tags are part of the taxonomy, which is invalid in the context of CTS since users tend to use tags in an ad-hoc manner, resulting in a substantial noise; (iv) There is no evaluation. Our approach addresses all these issues.

In [36], the authors adapt association rule mining to build a tree from the tags of a CTS. Since there can be only two dimensions for the association rules, they propose an approach for converting three dimensional data of (user, tag, resource) to two dimensional data, by projecting on the target dimensions. Using support and confidence measures from association rules, they filter the rules and build the tree. However, the authors do not explain how edges are built from association rules. They evaluate association rule mining algorithms on two forms of data: (user, tag) and (user, resource). In contrast, our association rule mining increases precision by using both confidence and reverse confidence; additionally we use two other steps to improve precision and recall.

[38] extends [36] and [22] by considering the tag's context. The authors argue that similarity between a tag and its parent should also depend on the parent's ancestors to avoid chains such as *design* $\rightarrow$ *web* $\rightarrow$ *howto* $\rightarrow$ *productivity* $\rightarrow$ *business*, where each edge makes sense but the complete chain does not. Here, $\rightarrow$ denotes the parent-child relationship with the left term as the parent. Evaluations are based on the structure of the tree such as the maximum and average depths of the tree. Barla and Bieliková [9] consider tag context similarly to [38]. For each tag, [9] finds the tag that co-occurs most frequently with it and creates a child-parent or sibling relationship between the two depending on the frequencies of the two tags. [9] assumes that each tag has at most one parent and potentially ignores some relationships between concepts.

In [14], the authors distinguish between subjective tags (e.g., "neat") and objective tags (e.g., "Mac"). The authors calculate feature vectors for each objective tag by Probabilistic Latent Semantic Indexing [23]. Then, their DAG algorithm calculates entropy values for each tag from feature vectors. The tags form a directed graph where tags with higher entropies are in higher levels of abstraction. The DAG algorithm also assumes that all objective tags are part of the taxonomy; this may not be valid for all domains. They consider an edge to be correct if there exists *any* relationship between concepts. Note that this definition yields an artificially higher precision compared to defining correctness of edges with regarding to specific relationships (e.g., is-a).

[27] focuses on categorizing users by the kind of tags that they use. They show that excluding some users can reduce noise and improve precision. We do not exclude any users but adapting TECTAS to the domains with lots of spam users is a future work.

Previous research in the area of exploiting lexico-syntactic patterns to detect relationships between terms was conducted by [21]. In [21] authors define a collection of patterns in text documents that indicate is-a relationship between words. Also, patterns for detecting has-a relationships from text corpora have been explored in the literature of natural language processing [11, 15]. Authors of [13] integrate the idea of using lexical patterns for detecting is-a relationships, and the idea of analyzing the number of matches for these patterns found in a search engine. Last phase of the TECTAS algorithm for detecting has-a relationships is different from previous approaches: (1) We detect has-a relationship between terms instead of is-a relationships (2) We extract has-a relationships from the domain of CTSs instead of text documents. To our knowledge, this combination has not been addressed in the literature.

Our approach is different and more robust than previous methods. First, we carefully chose measures for rule sets extracted by an association rule mining algorithm, which results in high precision. Second, the second stage of our algorithm prunes irrelevant edges between bi-gram elements very accurately. Third, we recover many edges where pairs of tags share the same child. Fourth, in addition to is-a relationships, our method is capable of detecting has-a relationships with high accuracy. Finally, we provide a detailed discussion of the metrics that can be applied to taxonomies generated from CTSs. These metrics both allow a detailed comparison of our work with previous work and pave the way for future algorithms to be evaluated more rigorously.

Chapter 4

Taxonomy Extraction from Collaborative TAgging Systems (TECTAS) Algorithm

Our algorithm for taxonomy extraction from CTSs is predicated on the hypothesis that users tend to assign both child and parent tags (from an underlying taxonomy in their mind) when tagging resources. We leverage association rules [6] between tags to create this taxonomy. However, applying association rules is not enough. First, tags are often bi-grams (e.g., "free software") or more generally *n*-grams, but the bi-grams are often represented as separate tags (e.g., "free" and "software"). This is caused by user tendencies and because some CTSs split bi-grams programmatically. Bi-grams can confuse any algorithm based on co-occurrence of words. Second, a high co-occurrence frequency in itself does not imply that the tags in question are involved in an is-a relationship. Third, co-occurrence is a symmetric relationship and by itself does not suggest a direction for the generalization/specialization relationship between a pair of tags. Finally, applying a high enough threshold to remove noisy data can result in the exclusion of valid is-a relationships.

As shown in Algorithm 4.1, the TECTAS algorithm consists of five phases: (1) preprocessing (2) detecting possible edges (3) pruning candidate edges (4) discovering more is-a relationships based on co-parents (5) detecting has-a relationships.

In this chapter of thesis these four stages are described.

Algorithm 4.1 TECTAS

```
Input: (D) A set of 2-tuples in form of $< item, tag >$ or 3-tuples in form of $< user, item, tag >$
Output: (T) Taxonomy of tags (H) A set of 2-tuples with has-a relationship
1: $D' \leftarrow$ Preprocess (D)
   /*$D'$ is a set of $< item, tag >$ tuples*/
2: $< T_{basic}, F > \leftarrow Association\_Rule\_Mining(D')$ /*Algorithm 4.2*/
   /* $F$ is the set of frequent itemsets*/
3: $T_{pruned} \leftarrow Bi-gram\_Filtering(T_{basic})$ /*Algorithm 4.3*/
4: $T_{co-parent} \leftarrow Co\_Parent\_Pruning(T_{pruned}, F)$ /*Algorithm 4.4*/
5: $T$ and $H \leftarrow Has-A\_Detection(T_{co-parent})$ /*Algorithm 4.5*/
6: Return $T$ and $H$
```

4.1 Preprocessing

The preprocessing step gets the data ready for the association rule mining algorithm; it is primarily a cleaning step. All of the steps are basic and could be applied to any algorithm for creating a taxonomy from tags in CTSs.

The preprocessing step takes as input the CTS i.e., a set $< user, item, tag >$ tuples, or a set $< item, tag >$ tuples if user information is unavailable. Since TECTAS does not use user information, any provided user information is projected away to form $< item, tag >$ tuples.

Next, non-English keywords are removed from the input data. This preprocessing step is the same as [14]. Words that contain any non-English characters are considered as non-English words. This was adequate to remove non-English words from all of our datasets; other datasets may require more complex preprocessing.

The next step performs basic stemming, such as substituting singular nouns for their plural forms. Then, tuples containing tags or items that are used very rarely are removed from the tuple set. If tags or items occur only a small number of times, extracting is-a relationships for those tags is not statistically reliable; hence, we remove tags or items that occur less than a threshold. We empirically set 15 as the threshold for removing rare tags and items.

Finally, tags in the form of verbs or verb phrases (e.g., "read", "read but not owned") are detected by applying the Stanford parser[1] to each tag. Verbs and verb phrases are removed because the taxonomy only consists of concepts. Thus tags for task organizing [16] which occur frequently but convey no meaning about the item being tagged, can be pruned effectively. Other preprocessing techniques such as eliminating infrequent tags cannot handle this situation.

The preprocessing phase outputs a set of 2-tuples in form of $< item, tag >$. It is possible for tuples to occur more than once, which means that different users have applied the same tag to an item.

4.2 Using Association Rule Mining to Detect Edges

Exploring Association Rule Mining Possibilities

To adapt the tagged data to market basket analysis, we must define how to build transactions from tag assignments. We explored three different definitions of "co-occurrence" to see how effective they were at detecting is-a relationships:

DEFINITION 1: Tags t and t' co-occur if the same user used both t and t' (possibly on different items). $freq(t,t')$ is the number of distinct users who used both t and t'.

DEFINITION 2: Tags t and t' co-occur if both were used to tag the same item (by possibly different users). $freq(t,t')$ equals the number of distinct items which were assigned both tags of t and t'.

DEFINITION 3: Tags t and t' co-occur if the same user used them on the same item. $freq(t,t')$ is the number of distinct $< user, item >$ pairs such that t and t' were assigned to that $item$ by the $user$.

In each of these three definitions, transactions are built differently. Next, we define a notion of tag frequency which is simply transactions containing the tag, under each notion of transaction. For Definition 1, freq(t) = # distinct users who used tag t; for Definition 2, freq(t) = # distinct items which received tag t; for Definition 3, freq(t) = # $< user, item >$ pairs such that $user$ tagged $item$ with tag t. Using these notions, all other measures such as support, confidence, etc. can be derived for each definition.

[1] http://nlp.stanford.edu/software/lex-parser.shtml

Empirical analysis revealed that Definition 1 is too tolerant — too many pairs of tags are labeled with is-a relationship, so it suffers from poor precision. Definition 3 is too conservative. By insisting that the same user tag the same item with both tags, too few genuine tag pairs are labeled with is-a relationship, so it suffers from poor recall. Therefore, TECTAS uses Definition 2.

TECTAS's Use of Association Rule Mining

As shown in Algorithm 4.2, TECTAS generates a set of possible edges by using tag co-occurrences. Tag co-occurrences have previously been used to derive concept hierarchies from text [35], induce an ontology from tag space [37] and cluster tags [10]. Our work is the first to consider the different kinds of co-occurrence (Section 4.2) and thus optimally use association rules to create a taxonomy from user tags. We use the FP-tree association rule mining algorithm [20] to extract frequent tag sets[2] and interesting rules from the set of transactions. The *support* of a tag set X is defined as the proportion of transactions containing tag set X and the *confidence* of a rule is defined as $\text{confidence}(X \Rightarrow Y) = \text{support}(X \cup Y)/\text{support}(X)$ — i.e., how often do X and Y occur together divided by how often X appears.

We assume users tend to tag an item with both a term in the taxonomy and its parent; we validate this assumption in Section 5.3. Therefore, if two keywords co-occur frequently, they are likely to be related with an is-a relationship. We use *support* to filter transactions with cardinality of two. However, popular unrelated terms may occur together in many transactions; so we use *confidence* to remove tuples containing unrelated tags.

Because terms which co-occur with high confidence are sometimes synonyms (e.g., “os” and “operating system”), we use confidence in the reverse direction to ensure that terms have different level of abstraction. We assume that general terms occur more frequently; as a result, TECTAS chooses the more frequent term as the generalization in the is-a relationship. We validate this in Section 5.3.

Table 4.1 defines the terms in TECTAS. Forward confidence is the confidence for the rule. Reverse confidence is the confidence of the rule in the reverse direction. Parameter values were chosen empirically: *min_support* is 2% of the number

[2]Tag sets correspond to itemsets in the context of frequent itemset mining.

Algorithm 4.2 Association_Rule_Mining

```
Input: (D) A set of 2-tuples in form of $< item, tag >$
Output: (T) Preliminary taxonomy of tags
Output: (F) Set of frequent itemsets
1: Group $D$ by $item$, collecting the set of tags associated with each $item$. /*create:
   $< item, \{tag_1, ..., tag_k\} >$*/
2: $S \leftarrow$ Union of tags associated with each item (i.e., S is set of transactions)
3: $F \leftarrow$ Frequent itemsets of size two from $S$ where support $> min\_support$ /*$FC_i$
   and $RC_i$ are respectively
   forward and reverse confidence for rule $i$.*/
4: for all $F_i \in F$ do
5:    if (($FC_i \geq min\_conf.$) and ($RC_i \leq 1 - min\_conf.$)) OR (($RC_i \geq min\_conf.$)
      and ($FC_i \leq 1 - min\_conf.$)) then
6:       Add $F_i$ to $T$ with more frequent tag as the more general term
7:    end if
8: end for
9: Return $< T, F >$
```

Table 4.1: TECTAS algorithm terms

Term	Definition
$\lvert T_x \rvert$	# of transactions containing tag T_x
$\lvert T_x.T_y \rvert$	# of transactions containing both T_x and T_y
$\lvert T_x.T_y \rvert / \lvert T_x \rvert$	Forward Confidence: conf. for rule $T_x \Rightarrow T_y$
$\lvert T_x.T_y \rvert / \lvert T_y \rvert$	Reverse Confidence: conf. for rule $T_y \Rightarrow T_x$

of all transactions and $min\_confidence$ is set to 70%.

4.3 Pruning Edges Between Bi-gram Elements

In this phase, edges between elements in a bi-gram (i.e., common phrase) are automatically pruned using a search engine. Tuples whose elements are bi-grams of words are pruned because they rarely indicate an is-a relationship. Usually bi-grams are compound nouns in the form of "adjective + noun" (e.g., free software) or "noun + noun" (e.g., web browser). These bi-grams do not contain is-a relationships but sometimes are detected in the edge detection algorithm as edges of a taxonomy since they co-occur.

The number of documents returned by search engines extracts the frequency of bi-grams in the corpus. Using a search engine to find bi-grams has been used in Natural Language Processing [25, 30, 44] but has not previously been applied to creating a taxonomy of tags from CTSs.

As shown in Algorithm 4.3, for each relationship tuple TECTAS sends two keyword queries to a search engine. The queries are the quoted permutations of the terms in the tuple. If the ratio of the number of results returned for the two queries is larger than a threshold, the terms in the relationship tuple are bi-grams. E.g., if the relationship tuple is $< software, free >$, the queries are “free software” and “software free”. Since the ratio is higher than the threshold for this tuple, it is detected as a bi-gram and pruned. The threshold for detecting bi-grams was computed experimentally. Because words in text documents have Zipfian distribution, [25] suggests using a logarithmic transformation of returned result counts. We found that the logarithmic transformation is also more accurate in detecting bi-grams.

Algorithm 4.3 Bi-gram_Filtering

```
Input: (T) A taxonomy of tags, which is a set of 2-tuples of the form < parentTag, childTag >
Output: ($T'$) A reduced taxonomy of tags
  $T' \leftarrow T$
  for all $T'_i \in T'$ do
    $order1 \leftarrow$ # of hits of querying “parentTag childTag” as a phrase
    $order2 \leftarrow$ # of hits of querying “childTag parentTag” as a phrase
    $ratio \leftarrow \frac{\log(\max(order1, order2))}{\log(\min(order1, order2))}$
    if $ratio \geq bi-gram\_threshold$ then
      remove $T'_i$ from $T'$
    end if
  end for
  Return $T'$
```

In this stage of the TECTAS algorithm, the precision of correct edges is increased, but recall is decreased; however, as shown in Section 5.4, the decrease in recall in negligible compared to increase in precision. If taxonomy building is done semi-automatically — where each discovered relationship must be validated by an

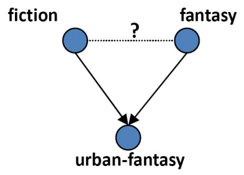


Figure 4.1: Concept of co-parents in a taxonomy

expert — lower thresholds should be used in this stage of the TECTAS algorithm. In the presence of an expert, higher recall is more desirable than the highest possible precision because an expect can reject a small number of incorrectly detected relationships without much effort.

4.4 Exploiting Co-parents to Find More Is-A Relationships

Examining the taxonomy built thus far reveals an interesting property when pairs of tags share the same child. For example, we may have "fiction $\rightarrow$ urban-fantasy" and "fantasy $\rightarrow$ urban-fantasy" in a taxonomy, where "fiction" and "fantasy" are both parents for "urban-fantasy" (Figure 4.1). However, the is-a relationship between "fiction" and "fantasy" is missing from our result. One possible reason for this omission is that people tend to use the more specific tags leading to "fiction $\rightarrow$ urban-fantasy" and "fantasy $\rightarrow$ urban-fantasy", so that "fiction $\rightarrow$ fantasy" does not occur above the relatively high threshold needed to avoid noise.

Hence we made the following conjecture: if a co-parent structure similar to the one in Figure 4.1 is more likely than usual, then the parents are in an is-a relationship. Hence, we added the following added step (Algorithm 4.4): for such co-parent pairs, we re-examine the forward-confidence and reverse-confidence under a lower threshold. If this pair passes the threshold, TECTAS adds an edge stating that the tag with higher frequency is a generalization on the other parent in the pair. In other words: we give tags in this co-parent structure a second chance.

As TECTAS's final step, if the taxonomy graph is disconnected, an "Entity" node is added as the root of taxonomy which is parent of all the nodes with no parents; this is common in other taxonomy finding algorithms as well.

Algorithm 4.4 Co_Parent_Pruning

```
Input: ($T$) A taxonomy of tags, which is a set of 2-tuples of the form $< parentTag, childTag >$
Input: ($F$) A set of frequent itemsets
Output: ($T'$) An enhanced taxonomy of tags
 1: $T' \leftarrow T$
 2: $G \leftarrow$ A directed acyclic graph constructed by edges from $parentTag$ to $childTag$ in every tuple of $T$.
 3: $S \leftarrow$ All tuples of tags $< parent_1, parent_2, child >$
    s.t. (1) edge($parent_1 \rightarrow child$) $\in G$ and
    (2) edge($parent_2 \rightarrow child$) $\in G$ and
    (3) edge($parent_1 \rightarrow parent_2$) $\notin G$ and
    (4) edge($parent_2 \rightarrow parent_1$) $\notin G$
 4: for all $S_i \in S$ do
 5:   for all $F_i \in F$ do
 6:     $F_i \leftarrow$ An itemset of size two including $parent_1$ and $parent_2$ in $S_i$
 7:     if ($parent_1$ and $parent_2 \in F_i$) AND $F_i \notin T'$ AND ((($FC_i \geq lower\_conf.$) and ($RC_i \leq 1 - lower\_conf.$)) OR (($RC_i \geq lower\_conf.$) and ($FC_i \leq 1 - lower\_conf.$))) then
 8:       /*$FC_i$ and $RC_i$ are respectively forward and reverse confidence for rule $i$.*/
 9:       Add $F_i$ to $T'$ with frequent tag as the more general term
10:     end if
11:   end for
12: end for
13: Return $T'$
```

4.5 Detecting Has-a Relationships

In this phase tuples that contain terms with has-a relationship between them are detected. The has-a relationships that we address in this thesis are more general than meronymy relationship which only reflects a physical point of view. The meronymy relationship subdivides concepts according to their components. According to [33], two general forms of meronymy can be defined: (1) Parts of structured objects e.g. the relationship between "Vancouver" and "Canada" (2) Parts of non-structured objects such as the relationship between "slice" and "pie" [42].

In our approach we extend has-a relationships to more abstract relationships

Table 4.2: Examples of terms with has-a relationship

Upper Class	Lower Class
human	body
human	mind
human	disease
google	googleMaps

which may not be covered by meronymy relationships. Table 4.2 shows some of the terms that we consider having has-a relationship. For two noun phrases NP_1 and NP_2 we consider them having has-a relationship (with NP_1 as the upper class or parent) if one of the following statements is true:

- NP_2 a is part of NP_1. E.g. “body” is a part of “human”.
- NP_1 has/have NP_2. E.g. “human” has “mind”.
- NP_1 may have NP_2. E.g. “human” may have “disease”.

Our approach to extract has-a relationships is based on a statistical analysis technique. We detect has-a relationships by analyzing occurrences of lexico-syntactic patterns in the web. Statistical analysis approaches which exploit lexico-syntactic patterns from text documents suffer from the sparse data problem [34]. Because of data sparsity, lexico-syntactic patterns do not occur frequent enough to detect accurate relationships between terms. Previous work [12] has shown that by using large amount of text, accuracy of statistical methods will be improved. [26] claims to solve sparse data problem by using web as the biggest data source available. Even though statistics from web are only based on indexed resources, authors of [13] claim to gain robust statistics by using web search engines.

According to previous methods (such as [11, 15]) in using lexico-syntactic patterns, several patterns for detecting has-a relationships have been proposed. Table 4.3 shows some of these patterns. Based on experimental results reported in [11, 15], patterns 1 and 2 are among the most common patterns for detecting has-a relationships. We used pattern 1 and also two variations of pattern 2 called pattern 2a and pattern 2b in detecting has-a relationships. These three patterns are shown

Table 4.3: Examples of lexical patterns indicating has-a relationships

	Pattern	Example
1	NP_1's NP_2	baby's mouth
2	NP_2 of a/an/the NP_1	ears of the baby
3	NP_1 have/has NP_2	table has legs
4	NP_{12}	door knob

Table 4.4: Lexical patterns used in TECTAS algorithm

	Pattern
1	NP_1's NP_2
2a	NP_2 of the NP_1
2b	NP_2 of NP_1

in Table 4.4. Even though using pattern 2b in Table 4.4 may seem unusual, we noticed that using this pattern will help in detecting more has-a relationships from the CTSs. For some tuples with has-a relationship between terms, frequency of pattern 2b is higher than pattern 2a. For example for the tuple <XmlHttpRequest, JavaScript>, the phrase "XmlHttpRequest of JavaScript" is more frequent than "XmlHttpRequest of the JavaScript".

Algorithm 4.5 determines whether two terms are related with a has-a relationship based on the number of matches returned from the search engine for the queries. Input tuples for Algorithm 4.5 are the tuples generated by Algorithm 4.4. Algorithm 4.5 starts by generating six keyword queries from each input tuple. For each pattern of Table 4.4 two queries are generated which are two quoted permutations of the terms in the tuple. Table 4.5 shows the generated queries for tuple <building, door> that are sent to the search engine.

In the next step, the ratio of hits is calculated for both permutations of each pattern. For example for pattern 1, two ratios called $p1\_ratio$ and $p1\_ratio\_Reverse$ are calculated using queries 1 and 2 from Table 4.5:

$$p1\_ratio = \frac{\text{\# of hits of Query 1}}{\text{\# of hits of Query 2}} \qquad p1\_ratio\_Reverse = \frac{\text{\# of hits of Query 2}}{\text{\# of hits of Query 1}}$$

If one of $p1\_ratio$ or $p1\_ratio\_Reverse$ is large enough, it may indicate that two

Table 4.5: Keyword queries generated in Algorithm 4.5 for tuple <building, door>

	Pattern	Query
1	1	building's door
2	1	door's building
3	2a	door of the building
4	2a	building of the door
5	2b	door of building
6	2b	building of door

terms have has-a relationship, but it is not enough. Similarly ratios for pattern 2a and 2b are calculated. The criteria of Algorithm 4.5 to announce a tuple with has-a relationship is as follows: the ratio for pattern 1 and either one of patterns 2a or 2b should be greater than a threshold. This threshold is called has-a_threshold and was computed experimentally and set to 8.

Our has-a detection algorithm is currently limited to extracting has-a relationships between single-word tags. In case of multi-word tags, such as <ancient Greek, trojan war>, the number of matches for associated queries are small; hence, Algorithm 4.5 can not detect has-a relationships. Adapting Algorithm 4.5 to multi-word tags is a future work.

Algorithm 4.5 Has-A_Detection

```
Input: (T) A taxonomy of tags, which is a set of 2-tuples of the form $< tag1, tag2 >$
Output: ($T'$) A reduced taxonomy of tags (H) A set of 2-tuples containing terms with has-a relationship
 1: $T' \leftarrow T$
 2: for all $T'_i \in T'$ do
 3:     $hits1 \leftarrow$ # of hits of querying “tag1 ’s tag2” as a phrase
 4:     $hits2 \leftarrow$ # of hits of querying “tag2 ’s tag1” as a phrase
 5:     $p1\_ratio \leftarrow \frac{hits1}{hits2}$
 6:     $p1\_ratio\_Reverse \leftarrow \frac{hits2}{hits1}$
 7:     $hits3 \leftarrow$ # of hits of querying “tag2 of the tag1” as a phrase
 8:     $hits4 \leftarrow$ # of hits of querying “tag1 of the tag2” as a phrase
 9:     $p2A\_ratio \leftarrow \frac{hits3}{hits4}$
10:     $p2A\_ratio\_Reverse \leftarrow \frac{hits4}{hits3}$
11:     $hits5 \leftarrow$ # of hits of querying “tag2 of tag1” as a phrase
12:     $hits6 \leftarrow$ # of hits of querying “tag1 of tag2” as a phrase
13:     $p2B\_ratio \leftarrow \frac{hits5}{hits6}$
14:     $p2B\_ratio\_Reverse \leftarrow \frac{hits6}{hits5}$
15:     $p2\_ratio \leftarrow \max(p2A\_ratio, p2B\_ratio)$
16:     $p2\_ratio\_Reverse \leftarrow \max(p2A\_ratio\_Reverse, p2B\_ratio\_Reverse)$
17:     if $((p1\_ratio \geq has-a\_threshold)$ and $(p2\_ratio \geq has-a\_threshold))$ then
18:         add $T'_i$ to $H$ with tag1 as upper class
19:         Remove $T'_i$ from $T'$
20:     else
21:         if $((p1\_ratio\_Reverse \geq has-a\_threshold)$ and $(p2\_ratio\_Reverse \geq has-a\_threshold))$ then
22:             add $T'_i$ to $H$ with tag2 as upper class
23:             Remove $T'_i$ from $T'$
24:         end if
25:     end if
26: end for
27: Return $T'$ and $H$
```

Chapter 5

Experiments

5.1 Datasets

We used four data sets in our experiments: Del.icio.us (a social bookmarking web service), IMDb (the Internet Movie Database), LibraryThing (for tagging books) and CiteULike (a service for storing, organizing, and sharing scholarly papers).

Table 5.1 compares the datasets. User information is not available in the IMDb dataset, so competing algorithms were unable to create taxonomies from it.

5.2 WordNet is Not Enough

We examined if creating a specialized taxonomy was necessary given readily available taxonomies such as WordNet; this section validates that WordNet is not enough because WordNet misses many relationships between terms even when it contains

Table 5.1: Corpus details in some collaborative tagging systems

	Del.icio.us (Dec. 2007)	CiteULike (Jan. 2010)	IMDb (Nov. 2009)	LibraryThing (corpus from Delft)
# of Tags	6,933,179	431,160	2,593,747	10,469
# of Items	54,401,067	2,081,799	356,162	37,232
# of Users	978,979	60,220	Not Available	7,279
# of Tag Assignments	450,113,886	7,922,454	2,625,237	2,415,517

Table 5.2: Precision for all relationships

	Del.icio.us (Aug. 2005)
Precision Examined by WordNet	0.25
Precision Examined Manually	0.66

Table 5.3: Precision for relationships with both terms in WordNet

	Del.icio.us (Aug. 2005)
Precision Examined by WordNet	0.46
Precision Examined Manually	0.63

both terms.

Tables 5.2 and 5.3 show the results of checking the output of TECTAS both manually and by using WordNet. In manual evaluation, each edge (relationship) in the taxonomy is checked, and if two terms in the edge have an is-a relationship, that edge is labeled as correct. Evaluation by WordNet is done as follows: for the child term in each relationship, parents of the child are determined by considering all senses (all meanings) of the child in WordNet. For each sense, all the parents of the child are checked recursively and added to the pool of parents for that child. Finally, for each child, if the parent term detected by TECTAS is in the pool of parents for that child, that relationship is labeled as correct.

Table 5.2 shows the results of validating all the relationships extracted by TECTAS on the Del.icio.us dataset. Next, we considered only relationships detected by TECTAS where both terms in the relationship existed in WordNet. Table 5.3 shows that there are many relationships between terms in WordNet that WordNet will not find. For example, WordNet contains 3 senses for “python”, but none of these senses is related to programming; as a result, “programming $\rightarrow$ python” is missing in WordNet.

Using Tables 5.2 and 5.3, the percentage of missing relationships in WordNet is calculated in Table 5.4. In this table, the percentage of missing edges for all relations validates the fact that WordNet does not contain all the terms used in

Table 5.4: Percentage of missing relationships in WordNet

	Del.icio.us (Aug. 2005)
all relationships	62.12%
relationships with both terms in WordNet	26.98%

Del.icio.us. The percentage where both terms are available in WordNet shows that WordNet not only misses many terms in its database, but also misses many edges between the terms that are available in its database.

5.3 Validating Assumptions in TECTAS

TECTAS is built on the hypothesis that users tend to tag items with both a term and its parent. This hypothesis is based on observing the tagged data and it is originated from the fact that when people naturally think of a word, the more generalized word of the original word comes to their minds. To verify this hypothesis, we looked at the tagging behavior of users throughout our datasets. Because it is impossible to manually validate millions of tag assignments, we needed to automate the process. We did so by checking for each tag t that has parents in the dataset, what percentage of times does t occur with one of its parents.

Next we must define the possible parents; if we define parents based only on what the algorithms have found, then we will miss those cases where there are parents that the algorithms have missed — which are likely to be the cases where the tags did not occur with their parents.

Therefore, for a given tag t, we used two metrics: S_t is the set formed by the union of the validated parent tags of t discovered by any of the algorithms. We refer to S_t as the *closed world parent set*. W_t is the set of tags such that each tag w in W_t is a parent of t according to WordNet, and w and t appear in the same dataset. The *open world parent set* is $S_t \cup W_t$. These terms indicate our belief that in S_t, the only parents that exist are those that have been found by some algorithm (hence it is a closed world) and that in W_t there may be other parent relationships that were not found (hence $S_t \cup W_t$ is an open world).

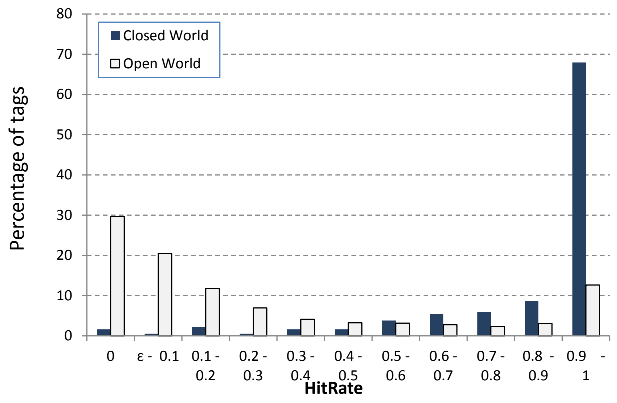


Figure 5.1: Histogram of HitRate values

We use the term "HitRate" to define how often tag t occurs with a member of its parent set divided by the total number of transactions containing t. Figure 5.1 shows a histogram of HitRate values; each histogram bucket shows the percentage of tags with this HitRate in both the closed and open worlds.

Figure 5.1 validates that people use parent and child tags together. In the closed world, more than 95% of child tags have a hit rate larger than 0.5, which means tags are usually accompanied with their parent tags in the same transaction. In the open world, we notice that 30% of tags have a hit rate of 0, which is also expected. We include parent words from WordNet for any semantic senses of a tag. It is highly possible that in the dataset the parent word does not have the corresponding meaning as it has in the is-a-relationship discovered by WordNet. It is worth noting two factors in this open world that strengthen our hypothesis: (1) more than 70% of tags occur with some parent and (2) more than 24% of child tags have a hit rate larger than 0.5. This is quite high, which validates our hypothesis that people tag items with both a tag and its parent, and explains why our recall is so high (as will be shown in Section 5.4).

Another assumption in TECTAS is that tags that occur frequently are the more general tags in the taxonomy, i.e., given two tags a and b in an is-a relationship, if a occurs more often than b, a is more general than b. We theorize that this is because more general tags can be related to more items than specific tags. To experimentally validate that this tagging behavior is true, we ran TECTAS on data of one month of the Del.icio.us dataset. We then built two taxonomies relying on frequency to determine which element was the parent and which was the child. The first taxonomy used the more frequent term as the generalization term in each edge, and the second taxonomy used the more frequent term as the specialization. Then we manually chose all the correct is-a relationships. The first taxonomy, which had the more frequent term as the generalization, validated this choice in TECTAS: 91% of the correct is-a relationships had the more frequent term as the generalization.

5.4 Performance Metrics

Having verified the necessity of creating a specialized taxonomy in Section 5.2 and that TECTAS's assumptions are valid in Section 5.3, we are almost ready to evaluate TECTAS's performance. First, however, we must consider the performance measures. We begin by considering all of the measures that we found for this problem in the literature:

1. Precision of is-a: the number of correctly detected is-a edges over the number of all the edges in the taxonomy.
2. Average number of children.
3. Maximum depth of the tree/DAG.
4. Average depth of tree/DAG.
5. Precision of any: the number of *any* correctly detected edges over the number of *all* the edges in the taxonomy. We use this metric because some of the algorithms do not distinguish relationship types.

For the metrics associated with depth or breadth, we calculate these metrics on a taxonomy with only correct relationships. Otherwise, algorithms with poor precision will appear to be finding more complex taxonomies when in fact they are simply finding more complex incorrect relationships.

Table 5.5: Evaluating the different phases of TECTAS

Dataset	Algorithm	Precision	Relative Recall	Avg. #	Avg. Depth	Max. Depth
Del.icio.us	BASIC	0.47	0.57	0.56	1.46	3
	PRUNED	0.59	0.52	0.59	1.54	3
	TECTAS	0.59	0.61	0.69	1.69	4
CiteULike	BASIC	0.25	0.32	0.26	1.22	3
	PRUNED	0.33	0.31	0.3	1.26	3
	TECTAS	0.33	0.34	0.32	1.3	3
IMDb	BASIC	0.32	N/A	0.4	1.33	2
	PRUNED	0.36	N/A	0.44	1.36	2
	TECTAS	0.38	N/A	0.5	1.42	3
LibraryThing	BASIC	0.39	0.46	0.46	1.36	2
	PRUNED	0.47	0.31	0.53	1.41	2
	TECTAS	0.49	0.34	0.58	1.66	4

We also propose a new metric: relative recall. It is impossible to find all of the is-a edges in the datasets that we tested, because there may be millions of tag assignments. Therefore, we compute the recall *relative* to the validated is-a edges found by all of the algorithms. For a given dataset, the relative recall for an algorithm is the number of valid is-a relationships found by that algorithm divided by the number of is-a relationships in the union of the results of all algorithms on that dataset.

5.5 Evaluating TECTAS's Phases

We first validated the various phases of TECTAS by comparing how three variations of it performed: BASIC uses only the base detection algorithm in Section 4.2. PRUNED includes both the basic algorithm and the pruning step in Section 4.3. TECTAS includes all of the steps in PRUNED plus the co-parent exploitation in Section 4.4. Table 5.5 shows evaluation results for different phases of TECTAS algorithm. Based on Table 5.5, even though the four datasets come from different CTSs, the results of experiments are very similar across the datasets. This suggests that TECTAS algorithm is applicable to other CTSs as well.

Table 5.6: Effect of co-parents

	Del.icio.us	LibraryThing	IMDb	CiteULike
Local Precision	61%	87%	75%	100%
Pattern Frequency	78%	77%	55%	20%

Effect of Pruning: Table 5.5 shows that for the Del.icio.us dataset, precision increases 22 percentage between BASIC and PRUNED. Similarly the decrease in recall for the same versions on Del.icio.us is 5 percentage points; the improvement in precision is higher than the decrease in recall by a factor of 4. As shown in Table 5.5, this behavior for precision and relative recall is fairly similar for all datasets. Note that the average number of children and average depth goes up at each step; this is because despite the fact that the pruning step removes some relationships, it prunes a lot of nodes that are leaves or have very few children.

Effect of Exploiting Co-parents:

We propose two metrics to validate that the co-parent pattern occurs:

$$\text{Local Precision} = \frac{(\text{\#correct edges})}{(\text{\#edges reported in this stage})}$$

$$\text{Pattern Frequency} = \frac{(\text{\#correct edges})}{(\text{\#pairs of co-parents})}$$

The local precision reflects the effectiveness of exploring co-parent structures, while the pattern frequency shows the percentage of co-parents that actually have a correct is-a-relationship between them, so it also includes the correct edges between co-parents detected before post-processing.

Table 5.6 shows the results on all four datasets. The local precision is high, which means the discovery of more relationships does no harm for overall precision, not to mention it can increase recall. The effect on precision and recall is further shown in Table 5.5. As we can see, despite the fact that we are adding more relationships, in each case the overall precision is either the same or improved slightly, while the relative recall improves more.

As shown in Table 5.6, the pattern frequency is also relatively high except for

CiteULike. This shows that the co-parent structure gives a good implication of is-a relationships in most cases. The possible reason that pattern frequency is not very high in CiteULike is that "all correct edges" refers to only those detected correct edges in TECTAS with this stage. Some pairs of co-parents may still fail to be chosen due to the threshold or they are just too general.

5.6 Evaluation of TECTAS in Detecting Has-A Relationships

In this section we evaluate our proposed method for detecting has-a relationships described in Section 4.5. Since other competing algorithms do not addresses the problem of detecting these relationships from CTSs, we only report precision and recall of our method in this section. Table 5.7 shows the result of applying Algorithm 4.5 to the Del.icio.us and CiteULike datasets.

Table 5.7: Precision and recall in detecting has-a relationships

	Del.icio.us	CiteULike
Precision	100%	80%
Recall	11.1%	36.3%

As shown in Table 5.7, our proposed method is fairly accurate in detecting has-a relationships. Our algorithm did not detect any incorrect has-a relationships in the Del.icio.us dataset. In the CiteULike dataset, our method detected a small number of incorrect tuples such as <human, middle>. We expected a low number of hits for queries such as "human's middle" and "middle of human"; however, these queries have high hits because phrases such as "human's middle finger" and "middle of human history" are frequent. High hits of these phrases is because finger is-a-part-of human, but our method detects middle is-a-part-of human.

Recall of our proposed method is relatively low for two following reasons:

- Users of CTSs tend to use abbreviations and acronyms when tagging. For example in Del.icio.us dataset, tags such as "os," "dev," "webdev," and "app" are among the most frequently used tags. On the other hand, web documents

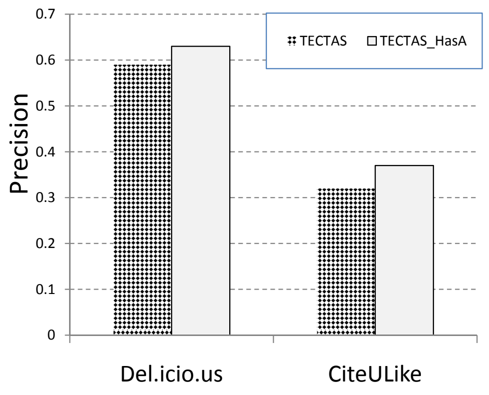


Figure 5.2: Effect of has-a detection phase in precision of is-a relationships

do not include these abbreviations and acronyms frequently enough for our method to detect has-a relationships.

- Users of CTSs use domain specific and uncommon words. For example consider the tuple <crystallography, model>. The word “crystallography” is frequently used in the CiteULike dataset but phrase “model of crystallography” occurs less than 100 times in all of the web documents. Despite the fact that model is-a-part-of crystallography, our method is unable to detect such a relationship.

The concern of using abbreviations and acronyms can be elevated by replacing acronyms and abbreviations with full word. For a tag such as “os” a dictionary of acronyms can be used to change it to “operating system”. For other tags such as “dev,” an expert should help with mapping this word to “development”. Detecting has-a relationships between abbreviations and acronyms is not addressed in this thesis and is a future work.

The has-a detection Algorithm introduced in Section 4.5 removes some tuples from the taxonomy of is-a relationships and labels those tuples with has-a relationship; hence, has-a detection phase will affect the precision and recall of detecting is-a relationships. In our experiments, Algorithm 4.5 removed some tuples that were incorrectly labeled with a is-a relationship. Figure 5.2 shows the effect of has-a detection phase in the precision of the TECTAS algorithm. In Figure 5.2, TECTAS refers to the TECTAS algorithm without using Algorithm 4.5 and TECTAS_HasA refers to the TECTAS Algorithm after Algorithm 4.5 is applied. The recall of detecting is-a relationships was not reduced after applying Algorithm 4.5 because no tuple with a correct is-a relationship was labeled with a has-a relationship.

As mentioned in Section 4.5, our proposed method for extracting has-a relationships is currently limited to detecting has-a relationships between single-word tags. For CTSs such as LibraryThing and IMDb where majority of tags are multiword tags, the recall of the algorithm drops drastically.

5.7 Comparing TECTAS to Other Algorithms

We compare our TECTAS algorithm with three algorithms from Chapter 3: 1) the DAG algorithm from [14] (hereafter "DAG-ALG") 2) Schmitz's algorithm [37] (hereafter "Schmitz"), and 3) Barla and Bieliková's algorithm [9] (hereafter "BB"). Since these algorithms cannot process the IMDb dataset due to the lack of user information, we compare only the Del.icio.us, LibraryThing, and CiteULike datasets.

To have a fair comparison, we implemented the above algorithms as closely as possible to the way that they were implemented by the authors; we used the parameters that were described in the papers and contacted the authors for additional information about how to make their algorithms as competitive as possible. TECTAS algorithm and four mentioned algorithms were implemented in the Java programming language.

In order to evaluate the edges manually, we needed the number of edges output for each algorithm to be small. We achieved this by putting another threshold on the number of times a tag, an item, or a user must occur in the dataset to be considered in the taxonomy. To be fair, we used the same threshold for each algorithm. Thus, there were fewer than 200 edges output by each algorithm.

In Figures 5.3- 5.8 results of TECTAS and three competing algorithms are shown. In this figure, comparison is done based on following metrics: (1) Precision: shows how many is-a relationships are correctly found. (2) Relative Recall:

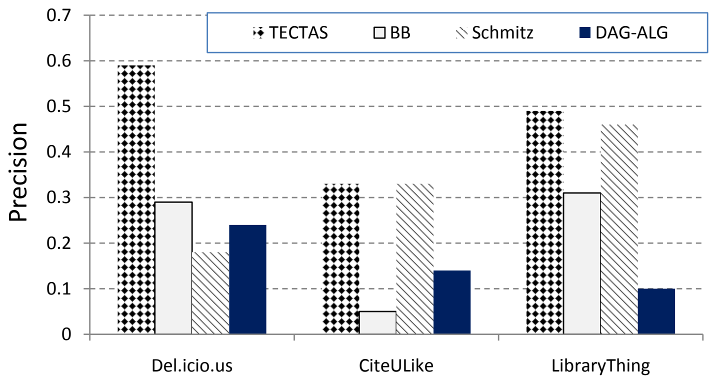


Figure 5.3: Precision of is-a relationships

shows which algorithm finds the most relationships, given a union of the validated set of is-a relationships found by each algorithm. (3) Average Number of Children: high average number of children, represents rich Tree/DAG. (4) Average Depth of the Tree/DAG: the deeper the average depth of the tree/DAG, the more complex the relationships. (5) Precision of All Relationships: some algorithms consider determining that two tags are related to be sufficient. By comparing algorithms for all relationships, we realized that TECTAS outperforms other algorithms even using this metric. (6) Maximum Depth: the deeper the maximum depth of the tree/DAG, the more complex the relationships.

Figure 5.3 shows the algorithms' precision; the precision of TECTAS is 0.59 for Del.icio.us, 0.49 for LibraryThing and 0.32 for CiteULike. TECTAS outperforms the precision of all other algorithms on all datasets. The primary competitor to our TECTAS algorithm is Schmitz. Schmitz lost many correct is-a edges since it builds a tree instead of a DAG; in some cases such as "win $\rightarrow$ win xp" and "software $\rightarrow$ win xp," Schmitz is forced to pick only one parent.

Figure 5.4 compares the relative recall for the four algorithms. Our TECTAS algorithm is the best algorithm for the Del.icio.us and LibraryThing dataset by far, and is the second best algorithm for the CiteULike dataset. One observation on the DAG-ALG algorithm is that by using the entropy measure, it detected many

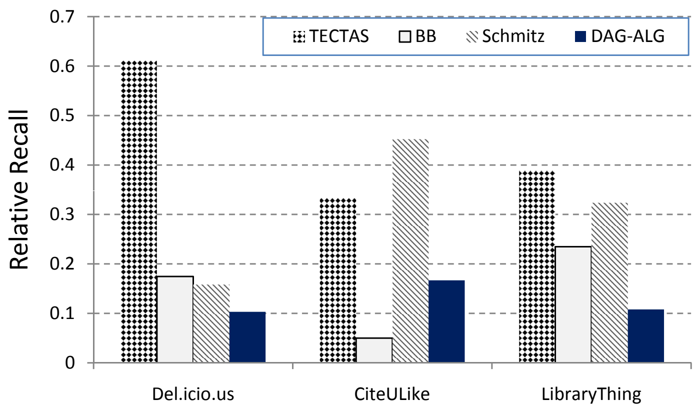


Figure 5.4: Relative recall

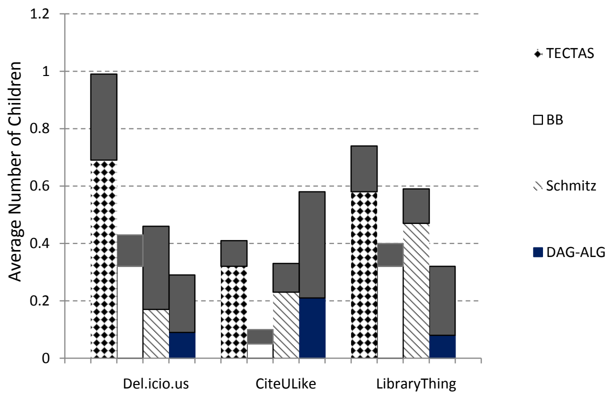


Figure 5.5: Average number of children

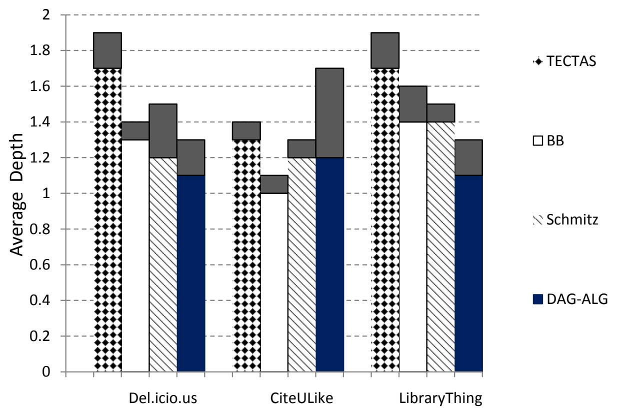


Figure 5.6: Average depth of the tree/DAG

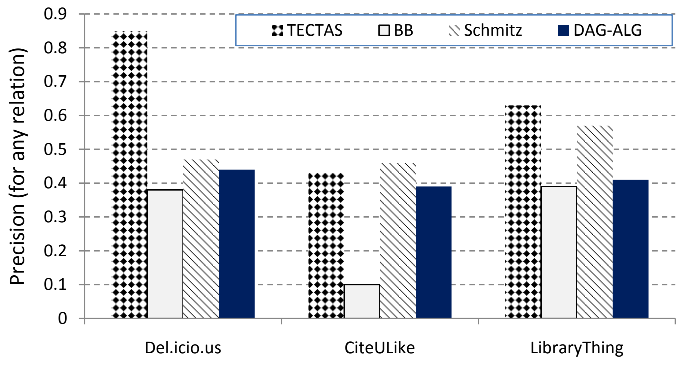


Figure 5.7: Precision of all relationships

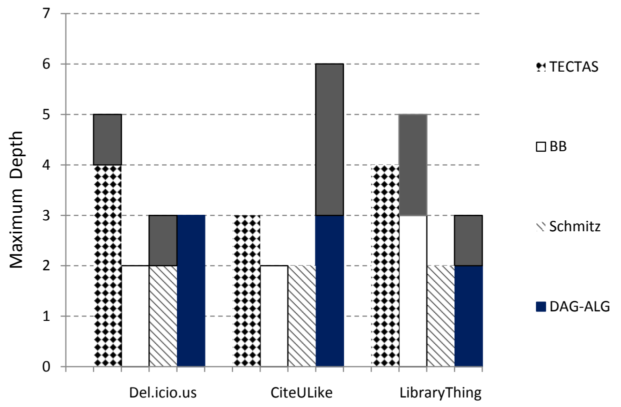


Figure 5.8: Maximum depth

Table 5.8: Percentage of relationships of various types

Dataset	Algorithm	% is-a	% sibling	% has-a	% overlap	% property	% ambiguous abbreviation	% reverse is-a
LibraryThing	TECTAS	78.30	0.94	0.94	9.43	10.38	0.00	0.00
	BB	70.83	12.50	4.17	1.39	6.94	0.00	4.17
	Schmitz	70.41	1.02	4.08	16.33	7.14	1.02	0.00
	DAG-ALG	24.73	20.43	8.60	5.38	17.20	5.38	18.28
Del.icio.us	TECTAS	73.01	6.72	8.94	0.00	8.45	0.96	1.92
	BB	75.86	0.00	10.34	0.00	10.34	0.00	3.45
	Schmitz	37.74	5.66	13.21	24.53	15.09	0.00	3.77
	DAG-ALG	31.71	14.63	21.95	4.88	21.95	0.00	4.88
CiteULike	TECTAS	79.28	2.86	3.58	2.80	5.74	0.00	5.74
	BB	45.45	9.09	9.09	9.09	9.09	0.00	18.18
	Schmitz	73.08	0.00	9.62	3.85	7.69	3.85	1.92
	DAG-ALG	35.00	0.00	10.00	15.00	35.00	0.00	5.00

popular tags in Del.icio.us such as "web," "software," etc. as subjective tags, and pruned them before trying to build the taxonomy edges, thus harming their relative recall.

Figures 5.5, 5.6, and 5.8 measure the depth and the number of children for the validated taxonomy detected by each algorithm. These measures quantify the richness of the taxonomy. In these three figures, lower bars show "is-a" relationships and higher bars show "any" relationships. For DAGs, if there are multiple paths from the root to a node *n*, we choose the depth as the longest path.

Because many of the other algorithms try to find "any" relationship between elements in the taxonomy, rather determining that the relationship is an is-a relationship, like TECTAS does, we measure both the "is-a" relationships that are found and "any" relationships found. As we can see, TECTAS has the highest maximum depth, average depth, and average number of children when is-a relationships are considered. When "any" relationships are allowed, TECTAS still performs fairly well. This is because there are so many is-a relationships detected as compared to the other relationship types. We analyzed the types of relationships that were found by all of the algorithms. We found that the "other" relationships fell into 6 different categories:

- **sibling**: the two tags are synonyms, antonyms, or counterparts of each other
- **has-a**: has-a or part-of
- **property**: one tag is used to describe another: e.g., "tool and mac", "tutorial and programming"
- **overlap**: there is some kind of connection or overlap between two tags but one is not wholly contained in another e.g., "fiction and American Literature", "folksonomy and classification".
- **ambiguous abbreviation**: ambiguous abbreviation: the tag is an ambiguous abbreviation, but the tags are likely to be related
- **reversed is-a**: the relationship is an is-a relationship, but it's reversed from the way that it is categorized

Table 5.8 shows the percentage of the different types of relationships as discovered by the various algorithms. As we can see, is-a is the only category of relationships that routinely has more than 20% of the relationships found by a given algorithm. Extending TECTAS to disambiguate between some of the other types of relationships is future work.

Finally, Figure 5.7 shows that our TECTAS algorithm outperforms other algorithms regarding precision for all relationships in two out of three datasets. Precision for all relationships is the measure used by DAG-ALG and BB to calculate the precision because their algorithms detect edges that are related in some way, but they do not distinguish the specific relationships between them.

Figures 5.3- 5.8 also show that most of the algorithms performed better on most measures for the Deli.icio.us and LibraryThing datasets. This validates the fact that the tags in these datasets are of better quality than the tags in CiteULike. Experimental results conducted on LibraryThing are reasonably good and can be improved given a larger dataset. This shows that we can compare different CTSs regarding the quality of tagging actions, using a taxonomy creation algorithm and possibly we can define some standards for CTSs e.g., if for a CTS the average number of children is greater than 3, that CTS has a *good* quality of tags.

In summary, our TECTAS algorithm outperforms three other algorithms for precision, maximum depth, average depth and average number of children, and produces the best relative recall and precision of all relationship in most cases. We realized that TECTAS is performing better based on not only the accuracy in detecting is-a edges, but also the richness of the resulting taxonomy.

Chapter 6

Conclusion and Future Work

6.1 Future Work

One extension of our TECTAS algorithm is to incorporate supervised learning to improve the quality of the extracted taxonomy. By adding a learning phase to TECTAS when edges are labeled by an expert, TECTAS will change the thresholds for confidence, reverse confidence, bigrams, etc, to discover more accurate relationships.

Finding relationships other than is-a and has-a will be another step toward building an ontology with more sophisticated relationships between concepts. As mentioned in Chapter 5, other interesting relationships between keywords include: synonyms, antonyms, and abbreviations.

Another extension of our proposed method as mentioned in Chapter 3 is to filter spam users. One of the challenges in extracting knowledge from CTSs is the huge number of automatically generated tags. These tags are mostly advertisement keywords or links to specific web sites as the keyword tags. By removing these users and tags associated with these users, TECTAS will create a more accurate taxonomy from the input CTS.

Adapting the TECTAS algorithm to the text domain can be another extension to this thesis. IR techniques are reasonably efficient and accurate in the domain of text; however, even in the text domain users are not usually familiar with the terminology of the corpus. For example, consider the case that a document collection contains multiple references to “OS” which is the abbreviation for “Operating System”. If a user searches for “Operating System”, he will not find any results in the corpus; however, if the search engine incorporates an ontology with synonymity and abbreviation relationships, a search engine will show the correct matches to the user.

We plan to work on new applications for ontologies built on top of collaborative tagging systems. One of the interesting applications is a query answering system which will use this ontology to reformulate the query and refine the results shown to the user. Consider the following example: a user is looking for information about different types of operating systems in a given document collection, and the search engine finds thousands of matches in the corpus. Finding an specific result item among this huge number of results is infeasible for the user. Hence, instead of showing numerous number of results, a subset of results can be selected by considering terms that are related to “operating system” in the ontology.

The proposed prototype mentioned above will be similar to the Clever Search system [28]. Clever Search enhances the search engines with information of word senses in WordNet. Clever Search exploits WordNet to reformulate queries by merging words and their word senses with AND operator. Our proposed prototype would be different in (1) exploiting specific relationships between words i.e. is-a and has-a relationships (2) relying on domain specific ontologies instead of WordNet.

Finally another future work is adapting generated ontologies to the SocialScope framework[7]. Information discovery layer in SocialScope analyzes the content to derive interesting new information. Information presentation layer of SocialScope explores the discovered information to build a dynamic result exploration framework. We are planning to embed our TECTAS algorithm in the information discovery layer of SocialScope and modify information presentation layer to exploit the domain specific ontologies generated by TECTAS.

6.2 Conclusions

In this thesis we introduced our TECTAS algorithm for extracting is-a and has-a relationships between tags from collaborative tagging system. Collaborative tagging systems contain non-structured data in nature, but in this thesis we showed that interesting relationships between keywords can be mined in these systems, resulting in a structured data of collaborative tagging systems.

The TECTAS algorithm introduced in this thesis uses association rule mining, bi-gram pruning, and exploiting pairs of tags with the same child to detect is-a relationships between tags. Also, a proposed method exploits lexical patterns and a search engine to detect has-a relationships. Details of each phase in the TECTAS algorithm was discussed in this thesis. Moreover, TECTAS can be used for extracting relationships automatically or semi-automatically. In the latter, an expert will help in verifying correctly detected relationships.

A prototype system based on TECTAS was developed to measure its accuracy and richness. A thorough analysis of TECTAS's assumptions, phases, and how our algorithm's performance is compared to other algorithms was reported in this thesis. In the process of evaluating algorithms, a couple of metrics previously addressed in the literature such as precision, average depth, and maximum depth was used. Also, we exploited the novel idea of using relative recall to evaluate the richness of algorithms.

Bibliography

[1] Wikipedia page for WordNet. http://en.wikipedia.org/wiki/Wordnet. → pages 5

[2] Wikipedia page for affinity analysis. http://en.wikipedia.org/wiki/Market_basket_analysis. → pages 6

[3] Wikipedia page for LibraryThing. http://en.wikipedia.org/wiki/LibraryThing. → pages 2

[4] Wikipedia page for Ontology Languages. http://en.wikipedia.org/wiki/Ontology_language. → pages 4

[5] Wikipedia page for ontology. http://en.wikipedia.org/wiki/Ontology-(computer-science). → pages 3

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994. ISBN 1-55860-153-8. → pages 7, 21

[7] S. Amer-Yahia, L. V. S. Lakshmanan, and C. Yu. Socialscope: Enabling information discovery on social content sites. In *Conference on Innovative Data Systems Research*, 2009. → pages 50

[8] Y. J. An, J. Geller, Y.-T. Wu, and S. A. Chun. Automatic generation of ontology from the deep web. In *International Conference on Database and Expert Systems Applications*, pages 470–474. IEEE, 2007. ISBN 0-7695-2932-1. → pages 17

[9] M. Barla and M. Bieliková. On deriving tagsonomies: Keyword relations coming from crowd. In *International Conference on Computational Collective Intelligence*, pages 309–320, 2009. → pages 19, 42

[10] G. Begelman, P. Keller, and F. Smadja. Automated tag clustering: Improving search and exploration in the tag space. In *Collaborative Web Tagging Workshop at WWW*, 2006. → pages 24

[11] M. Berland and E. Charniak. Finding parts in very large corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 57–64, Morristown, NJ, USA, 1999. Association for Computational Linguistics. ISBN 1-55860-609-3. → pages 19, 29

[12] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-intensive question answering. In *Information Technology: The Tenth Text Retrieval Conference (TREC 2001)*, pages 393–400, 2002. → pages 29

[13] P. Cimiano and S. Staab. Learning by googling. *SIGKDD Explorer Newsletter*, 6(2):24–33, 2004. → pages 19, 29

[14] T. Eda, M. Yoshikawa, T. Uchiyama, and T. Uchiyama. The effectiveness of latent semantic analysis for building up a bottom-up taxonomy from folksonomy tags. *World Wide Web*, 12(4):421–440, December 2009. → pages 19, 22, 42

[15] R. Girju, A. Badulescu, and D. Moldovan. Automatic discovery of part-whole relations. *Comput. Linguist.*, 32(1):83–135, 2006. ISSN 0891-2017. → pages 19, 29

[16] S. Golder and B. A. Huberman. The structure of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, 2005. → pages 23

[17] T. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *Interanational Journal of Human and Computer Studies*, 43(2):907–928, 1994. → pages 3

[18] J. Han and M. Kamber. *Data Mining: Concepts and Techniques (2nd Edition)*. Morgan Kaufmann Publishers, 2006. → pages vii, 5, 6, 7, 11, 12, 13, 14, 15

[19] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, pages 1–12, 2000. ISBN 1-58113-217-4. → pages 7

[20] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and*

Knowledge Discovery, 8(1):53–87, January 2004. ISSN 1384-5810. → pages 24

[21] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545, Morristown, NJ, USA, 1992. Association for Computational Linguistics. → pages 19

[22] P. Heymann and H. Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford University, April 2006. → pages 18, 19

[23] T. Hofmann. Probabilistic latent semantic analysis. In *Uncertainty in Artificial Intelligence, UAI'99*, pages 289–296, 1999. → pages 19

[24] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. BibSonomy: A social bookmark and publication sharing system. In *Conceptual Structures Tool Interoperability Workshop at the International Conference on Conceptual Structures*, 2006. → pages 3

[25] F. Keller and M. Lapata. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484, 2003. ISSN 0891-2017. → pages 26

[26] F. Keller, M. Lapata, and O. Ourioupina. Using the web to overcome data sparseness. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 230–237, Morristown, NJ, USA, 2002. Association for Computational Linguistics. doi:http://dx.doi.org/10.3115/1118693.1118723. → pages 29

[27] C. Körner, D. Benz, A. Hotho, M. Strohmaier, and G. Stumme. Stop thinking, start tagging: tag semantics emerge from collaborative verbosity. In *WWW*, pages 521–530, 2010. → pages 19

[28] P. M. Kruse, A. Naujoks, D. Rsner, and M. Kunze. Clever search: A wordnet based wrapper for internet search engines. In *Proceedings of the 2nd GermaNet Workshop*, 2005. → pages 50

[29] D. Laniado, D. Eynard, and M. Colombetti. Using wordnet to turn a folksonomy into a hierarchy of concepts. In *Semantic Web Application and Perspectives - Fourth Italian Semantic Web Workshop*, pages 192–201, Dec. 2007. → pages 17

[30] M. Lapata and F. Keller. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2:1–31, 2005. → pages 26

[31] H. Lin, J. Davis, and Y. Zhou. An integrated approach to extracting ontological structures from folksonomies. In *European Semantic Web Conference (ESWC)*, pages 654–668, 2009. → pages 17

[32] P. Mika. Ontologies are us: A unified model of social networks and semantics. *Web Semantics*, 5(1):5–15, 2007. ISSN 1570-8268. → pages 18

[33] S. Pribbenow. Meronymic relationships: From classical mereology to complex part-whole relations. *Green, R.; Bean, C.A.; Myaeng, S.H. : The Semantics of Relationships*, pages 35–50, 2002. → pages 28

[34] D. Sánchez and A. Moreno. Learning non-taxonomic relationships from web documents for domain ontology construction. *DKE*, 64(3):600–623, 2008. ISSN 0169-023X. → pages 17, 29

[35] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *SIGIR*, pages 206–213, 1999. ISBN 1-58113-096-1. → pages 24

[36] C. Schmitz, A. Hotho, R. Jäschke, and G. Stumme. Mining association rules in folksonomies. In *IFCS Conf., Studies in Classification, Data Analysis, and Knowledge Organization*, pages 261–270. Springer, 2006. → pages 3, 18, 19

[37] P. Schmitz. Inducing ontology from flickr tags. In *Collaborative Web Tagging Workshop at WWW*, 2006. → pages 17, 24, 42

[38] E. Schwarzkopf, D. Heckmann, D. Dengler, and E. Krner. Mining the structure of tag spaces for user modeling. In *Workshop on Data Mining for User Modeling*, 2007. → pages 19

[39] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007. ISBN 978-1-59593-654-7. → pages 5, 17

[40] G. O. Webb. An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 2000. → pages 7

[41] A. Weichselbraun, G. Wohlgenannt, A. Scharl, M. Granitzer, T. Neidhart, and A. Juffinger. Discovery and evaluation of non-taxonomic relations in domain ontologies. *Internationl Journal of Metadata Semantics and Ontologies*, 4(3):212–222, 2009. ISSN 1744-2621. → pages 17

[42] K. Weller and W. Stock. Transitive meronymy. automatic concept-based query expansion using weighted transitive part-whole-relations. *Information Wissenschaft und Praxis*, 59(3):165–170, December 2008. → pages 28

[43] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12:372–390, 2000. → pages 7

[44] X. Zhu and R. Rosenfeld. Improving trigram language modeling with the world wide web. In *Acoustics, Speech, and Signal Processing*, pages 533–536, 2001. → pages 26