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

Ali Moosavi, Tianyu Li, Laks V.S. Lakshmanan, and Rachel Pottinger

University of British Columbia, Vancouver, BC, Canada
{amoosavi,lty419,laks,rap}@cs.ubc.ca

Abstract. Ontologies define a set of terms and the relationships (e.g., IS-A and HAS-A) between them. An ontology relating the tags in a collaborative tagging system (CTS) makes the CTS easier to understand. We propose an algorithm to automatically construct an ontology from CTS data and empirically compare with related work on four real data sets – Del.icio.us, LibraryThing, CiteULike, and IMDb. We also verify the effectiveness of our algorithm in detecting IS-A and HAS-A relationships.

Keywords: ontology, taxonomy, tag, collaborative tagging systems

1 Introduction

Ontologies organize information in content management systems and are the core building blocks of the emerging Semantic Web. Substantial work has been done in extracting ontologies automatically from large repositories like text corpora, databases, and the web. This paper focuses on collaborative social tagging systems (CTSs) such as Del.icio.us (for tagging bookmarks), Flickr (for tagging photos), IMDb (for tagging movies), LibraryThing (for tagging books) and CiteULike (for tagging publications). These systems permit users to tag and share resources (documents, photos, videos, etc.). Our goal is to create a generic ontology of the tags from a CTS. By ontology, we mean a set of concepts from a domain, represented by the tags, and their (IS-A and HAS-A) relationships.

Learning an ontology from a CTS can help make the CTS more useful. For example, browsing an ontology of tags from a CTS can help users better refine their queries, either to find more items by using a more general term or to find fewer items by using a more specific term. This is especially important in a CTS since the resources are typically labeled by a small, sparse, set of tags — so discovering content in CTSs by simple keyword search is much harder than in document and web search. Another application of domain specific ontology builders is to enhance search engines with ontologies. E.g., the prototype Clever Search system [**?**] merges words and their word senses in the general ontology, WordNet [1], and returns more relevant result items to the user.

[1] http://wordnet.princeton.edu

In principle, we could use a general purpose ontology such as WordNet to browse a CTS; there are two disadvantages. 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, e.g., "Mac OS X". Secondly, as we demonstrate in Section 7, even when terms corresponding to tags in a CTS *are* present in WordNet, in many cases, valid IS-A relationships between them that are found by our algorithm are missing in WordNet. This mirrors a similar finding for the ontology extracted from Wikipedia using YAGO [**?**]; using a combination of WordNet and Wikipedia found significantly more ontological relationships (including IS-A) that were absent in WordNet.

This paper studies the following problem: given a collaborative tagging system consisting of users, resources (also called items), and tags assigned by users to items, extract an ontology consisting of tags in the CTS and IS-A and HAS-A relationships between the tags. We consider HAS-A relationships in addition to IS-A: indeed, IS-A and HAS-A relationships are among those most used in ontologies with rich relationships, such as WordNet.

Our algorithm for ontology extraction from CTSs is predicated on the hypothesis that tags assigned to a resource by a group of users tend to contain both child and parent tags. A possible explanation for this phenomenon is that different users may use tags at different levels of abstraction (from an underlying ontology in their mind); thus tags for the same item may include more abstract or more specific terms as an aggregation effect of various tagging behaviors. We leverage this hypothesis using association rules [**?**] and lexico-syntactic patterns to find relationships between tags. Our approach accounts for bi-grams (which can affect the precision of detected relationships), multi-word tags, and also infer non-trivial IS-A relationships from detected ones. We make the following contributions:

- We propose (Sections 4 through 6) an algorithm for ontology extraction from a CTS, called ONTECTAS (for ONTology Extraction from Collaborative TAgging Systems). The highlights of the algorithm include:
- We thoroughly explore how to use association rule mining over CTS tags to extract an ontology and show that one form of association mining significantly outperforms the others.
  - Candidate IS-A relationships are mined using association rules, making use of both forward and reverse confidence (Section 4.2).
  - Invalid tuples are pruned based on discovering bi-grams (Section 4.3).
  - Headword detection is leveraged for discovering relationships between multi-word tags (Section 4.4).
  - Lexico-syntactic patterns are used for detecting IS-A and HAS-A relationships. To our knowledge, we are the first to explicitly extract HAS-A relationships from CTSs (Section 5).
  - Based on items in the ontology having a common (IS-A) child, additional IS-A relationships are inferred (Section 6).
- We demonstrate via a comprehensive set of experiments on four real datasets that our algorithm outperforms previous algorithms w.r.t. quality and richness of the extracted ontology (Section 7).

– In addition to showing the effectiveness of ONTECTAS compared to previous works, our experiments confirm the following: (i) general purpose ontologies like WordNet miss many valid IS-A relationships (ii) the hypothesis that the population of users tend to use both parent and child tags (from an ontology in their mind) when tagging resources.

Section 2 discusses related work. Section 3 formalizes the problem studied in this paper. Section 8 concludes and discusses future work.

2 Related Work

A number of works focus on extracting ontologies from the web, e.g., in [**?**,**?**], the authors learn non-taxonomic relationships (e.g., "cures") from web documents. In [**?**], the authors complement general purpose ontologies such as WordNet by leveraging Wikipedia. In contrast, this paper focuses on CTSs and on extracting IS-A and HAS-A relationships.

Some other works have studied extracting ontologies from CTSs. Some approaches [**?**,**?**,**?**] match CTS tags to 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 are adhoc and use terms dynamically, general purpose ontologies miss many terms as well as edges (i.e., relationships). For example, our experiments show that WordNet misses more than 25% of correct edges between concepts extracted from Del.icio.us, *even when both parent and child concepts are in WordNet.*

Schmitz [**?**] 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 and HAS-A) between terms — it simply says they are related, not how. By contrast, our algorithm pinpoints IS-A and HAS-A relationships between terms.

Heymann and Garcia-Molina [**?**] create an ontology 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. Their method has following limitations: (i) sparseness of tag vectors causes tags around the leaves to be far from the topic of the root; (ii) they assume all tags are part of the ontology, which is invalid in the context of CTS since users tend to use tags in an adhoc manner, resulting in a substantial noise; (iii) There is no evaluation. Our approach addresses all these issues.

Schmitz et al. [**?**] use association rule mining to build a tree of related tags from a CTS; however, they do not explain how the edges are built or what types of relationships they model. We explain this in depth and also use lexico-syntactic patterns and a search engine to detect accurate IS-A and HAS-A relationships. [**?**] extends [**?**] and [**?**] by considering the tag's context. Barla and Bieliková [**?**]

consider tag context similarly to [**?**]. For each tag, [**?**] 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. [**?**] assumes that each tag has at most one parent.

In [**?**], 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 [**?**]. 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 ontology; 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 w.r.t. specific relationships (e.g., IS-A and HAS-A).

Lin et al. [**?**] build a subsumption graph from the folksonomy and use a random walk to sort tags by generality ranking. They put tags in the taxonomy based on support and confidence between candidate nodes from the graph. They only consider a single sense for each tag, which leads to missed relationships. The authors claim building transactions for tags associated to items by specific users will lead to the best taxonomy because it preserves most of the information. In contrast, we found that user information does not improve taxonomy quality.

Körner et al. [**?**] categorize users by the kind of tags they use. They show that excluding some users can reduce noise and improve precision. This improvement is orthogonal to the contribution we make in this paper and is applicable in our context as well. We leave adapting ONTECTAS to this as future work.

Hearst [**?**] defines a set of patterns that indicate IS-A relationships between words in text documents. [**?**,**?**] find patterns for detecting HAS-A relationships from text corpora. To our knowledge, our work is the first to extend the lexico-syntactic patterns to find relationships of any type between tags in CTSs.

In sum, in contrast to previous works on ontology extraction from CTSs, our method is capable of detecting both HAS-A and IS-A relationships and explicitly identifying each. Our multi-stage algorithm also extracts high quality relationships between multi-word tags.

3 Problem Statement

A *collaborative tagging system* [**?**] 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 by users, 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 CTSs may vary in detail from our definition above, e.g., IMDb does not have user information. We can model such CTSs by dropping U and defining $Y \subseteq T \times I$ as a binary relation. CTSs such as [**?**] allow users to declare their own IS-A relationships. User-supplied IS-A relationships can augment those automatically extracted but cannot supplant them because of the scale.

This paper studies how to efficiently extract IS-A and HAS-A relationships between tags in a given CTS. The output ontology consists ⟨tag1, tag2, label⟩

tuples where tag1 is the super class and label is either IS-A or HAS-A.[2] E.g., the tuple ⟨OS, Windows, IS-A⟩ indicates that Windows a kind of OS.

We want our algorithm to 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 relationships is not available beforehand and are too many to find manually. Thus, we want to use a "relative" notion of recall (defined in Section 7) and want to achieve a high score on it. Finally, we would like the extracted ontology's richness in terms of maximum depth, average number of children, and average depth to be high.

4 Ontology Extraction from Collaborative TAgging Systems (ONTECTAS) Algorithm

Algorithm 1 ONTECTAS

Input: (D) A set of $\langle item, tag \rangle$ 2-tuples or $\langle user, item, tag \rangle$ 3-tuples
Output: (O) Ontology of tags with IS-A and HAS-A relationships

1: $D' \leftarrow$ Preprocess D. /*D' is a set of $\langle item, tag \rangle$ tuples*/
2: $\langle T_{basic}, F \rangle \leftarrow Association\_Rule\_Tuple\_Detection(D')$ /*Algorithm 2*/
3: $T_{pruned} \leftarrow Bigram\_Filtering(T_{basic})$ /*Algorithm 3*/
4: $\langle T_{headword}, O \rangle \leftarrow Headword\_Detection(T_{pruned})$ /*Algorithm 4*/
5: $O \leftarrow O \cup$ IS-A$_Relationship_Detection(T_{headword},$ IS-A-patterns, IS-A $- threshold)$ /*Algorithm 5*/
6: $O \leftarrow O \cup$ HAS-A$_Relationship_Detection(T_{headword},$ HAS-A-patterns, HAS-A $- threshold)$ /*Algorithm ??*/
7: $T_{co\_parent} \leftarrow Co\_Parent\_Pruning(T_{headword}, F)$ /*Algorithm 6*/
8: Return $O \cup$ IS-A$_Relationship_Detection(T_{co_parent},$ IS-A $- patterns,$ IS-A $- threshold)$ /*Algorithm 5*/

Our ONTECTAS algorithm for ontology extraction (Algorithm 1) consists of six phases. First, data is preprocessed and cleaned. Next, we extract candidate tag tuples via association rule mining using forward and reverse confidence. We then remove tuples corresponding to bigrams. Next, we detect headwords of multi-word tags and use this to infer additional IS-A relationships. We then use lexico-syntactic patterns to extract additional IS-A and HAS-A relationships. Finally, we leverage pairs of tags sharing a common child in the extracted ontology to infer additional IS-A relationships. The next three sections describe the phases.

4.1 Preprocessing

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

[2] In both relationships tag2 IS-A tag1 and tag1 HAS-A tag2, we refer to tag1 as the super class label or the parent label for convenience, by abusing terminology.

Next, non-English keywords are removed from the input data. This preprocessing step is the same as as [**?**]. Words that contain any non-English character 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. If tags or items occur only a small number of times, extracting relationships for those tags is not statistically reliable; hence, we remove tags or items that occur less than a threshold. We empirically set 5 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[3] to each tag. Verbs and verse phrases are removed because the ontology only consists of concepts. Thus tags for task organizing [**?**] which occur frequently but convey no meaning about the item being tagged, can be pruned effectively.

The preprocessing phase outputs a set of 2-tuples in form of $\langle item, tag \rangle$. 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 Exploring Association Rule Mining Possibilities

Adapting tagged data to market basket analysis requires defining how to build transactions from tags, which in turn requires defining "co-occurrence". We explored three different definitions of co-occurrence:

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 $\langle user, item \rangle$ 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) = # $\langle user, item \rangle$ 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.

Empirical analysis revealed that Definition 1 is too tolerant — too many pairs of tags are labeled as in IS-A relationships, so it suffers from poor precision.

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

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 as in an IS-A relationship, so it suffers from poor recall. Therefore, ONTECTAS uses Definition 2.

ONTECTAS's Use of Association Rule Mining Algorithm 2. generates a set of possible edges by using tag co-occurrences.

While tag co-occurrences have previously been used to derive concept hierarchies from text [**?**], induce an ontology from tag space [**?**] and cluster tags [**?**]. Our work is the first to consider the different kinds of co-occurrence and thus optimally use association rules to create an ontology from user tags. We use the FP-tree association rule mining algorithm [**?**] to extract frequent tag sets[4] and interesting rules from the set of transactions. The *support* of a tag set X is the proportion of transactions containing tag set X and the *confidence* of a rule is defined as confidence$(X \Rightarrow Y)$ = support$(X \cup Y)$/support(X) — i.e., the proportion of transactions in which X and Y occur together among those in which X appears. In this paper, we refer to the well-known definition of confidence as forward confidence (FC). We also introduce a new notion, reverse confidence (RC) as follows: reverse_confidence$(X \Rightarrow Y)$:= support$(X \cup Y)$/support(Y).

Algorithm 2 Association_Rule_Tuple_Detection

Input: (D) A set of 2-tuples in form of $\langle item, tag \rangle$
Output: (T) Preliminary tag tuples, (F) Set of frequent itemsets
1: Group D by item. /*create: $\langle item, \{tag_1, ..., tag_k\}\rangle$*/
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 forward and reverse confidence respectively*/
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
7: **end if**
8: **end for**
9: Return $\langle T, F \rangle$

We assume users tend to tag an item with both a term in the ontology and another term that has relationship with it; we validate this assumption in Section 7.2. Therefore, if two keywords co-occur frequently, they are likely to be related. We use *support* to filter sets of tags with a cardinality of two. However, popular unrelated terms may occur together frequently, 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 are

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

related with IS-A or HAS-A relationships. Different values for *min_support* and *min_conf.* can drastically change the size of the ontology; in our experiments these values were chosen empirically. At the end of this step, we have not yet classified the relationships into IS-A and HAS-A.

4.3 Pruning Edges Between Bi-gram Elements

In this phase, bi-gram tuples which are common phrases are automatically pruned using a search engine. Usually bi-grams are compound nouns in the form of "adjective + noun" (e.g., free software) or "noun + noun" (e.g., web browser). Bi-grams do not contain IS-A or HAS-A relationships but sometimes are incorrectly detected as edges of an ontology since they co-occur frequently.

Finding bigrams by using a search engine [**?**,**?**,**?**] has not previously been applied to extracting relationships between CTS tags. ONTECTAS sends two keyword queries to a search engine for each relationship tuple (Algorithm 3). 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 regarded as bi-grams. E.g., if the relationship tuple is $\langle software, free \rangle$, 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. We experimentally found that the optimal threshold for detecting bi-grams is between 50 and 100. Because words in text documents have Zipfian distribution, [**?**] suggests using a logarithmic transformation of returned result counts. We found that the logarithmic transformation is also more accurate in detecting bi-grams.

Algorithm 3 Bi-gram_Filtering

Input: (T) A set of 2-tuples of the form $\langle tag1, tag2 \rangle$
Output: (T') A reduced set of 2-tuples
1: $T' \leftarrow T$
2: **for all** $T'_i \in T'$ **do**
3: $ratio1 \leftarrow$ # of hits of querying "$tag1$ $tag2$" as a phrase
4: $ratio2 \leftarrow$ # of hits of querying "$tag2$ $tag1$" as a phrase
5: $ratio \leftarrow \frac{\log(\max(ratio1, ratio2))}{\log(\min(ratio1, ratio2))}$
6: **if** $ratio \geq bi - gram\_threshold$ **then**
7: remove T'_i from T'
8: **end if**
9: **end for**
10: Return T'

At the end of bigram filtering, clearly the precision will be improved. A natural question is what is the price paid in terms of drop in recall. As shown in Section 7.5, we found that the decrease in recall in negligible compared to increase in precision.

4.4 Detecting Headwords in Multi-Word Tags

Since many CTS tags are multi-word tags in form of compound phrases such as "science-fiction" and "object-oriented-data-model", we use headword detection to extract additional IS-A relationships (Algorithm 4). First, the Stanford parser detects the *headwords* for each phrase. A headword is a phrase's grammatically most important word; it determines the phrase's syntactic type. We then extract an IS-A relationship for each multi-word tag by putting the headword as the parent of the whole phrase. E.g., we can infer "object-oriented data model" IS-A "model". In this phase, more candidate tuples are produced by using either whole phrases or their headwords as the tags in tuples.

Algorithm 4 Headword_Detection

Input: (T) A set of 2-tuples of the form $\langle tag1, tag2 \rangle$
Output: (T') A set of enhanced 2-tuples, (O) Ontology with IS-A relationships
1: $T' \leftarrow T$
2: **for all** $T_i \in T$ **do**
3: **if** T_i contains multi-tags **then**
4: $head1 \leftarrow$headword in $tag1$
5: $head2 \leftarrow$headword in $tag2$
6: $O \leftarrow O \cup \{\langle$head1, tag1, IS-A$\rangle\}$
7: $O \leftarrow O \cup \{\langle$head2, tag2, IS-A$\rangle\}$
8: $T' \leftarrow T' \cup \{\langle$head1, tag2$\rangle, \langle$head2, tag1$\rangle, \langle$head1, head2$\rangle\}$
9: **end if**
10: **end for**
11: Return $\langle T', O \rangle$

5 Using Lexico-Syntactic Patterns

Finally, we analyze occurrences of lexico-syntactic patterns to detecting IS-A and HAS-A relationships. Due to data sparsity, lexico-syntactic patterns do not occur frequently enough to accurately detect relationships between terms [**?**]. Previous work [**?**] has shown that by using large amount of text, accuracy of statistical methods will be improved. Hence, we build on [**?**] and query the web for more occurrences of the patterns. Even though statistics from the web are only based on indexed resources, [**?**] claim to gain robust statistics by using web search engines.

The core of our lexico-syntactic search is shown in lines 3-6 of Algorithm 5: given two tags and a pattern, we generate two keyword queries by considering the two possible permutations of the tags in the pattern. E.g., given ("human", "body", "'s"), the two generated queries will be "human's body" and "body's human". Then, the ratios for both forward and reverse occurrences direction are calculated. It is clear that given any set of patterns for any relationship, this algorithm can be applied. We use the following patterns from [**?**] to identify IS-A

Algorithm 5 IS-A_Relationship_Detection

Input: (T, P, threshold) Where T is a set of $\langle tag1, tag2\rangle$ tuples , P is a set of patterns
Output: (I) A set of 2-tuples in form of ⟨parent_tag, child_tag⟩

for all $t_i \in T$ **do**
  for all $p_j \in P$ **do**
    $hits1 \leftarrow$ # of hits of querying "$t_i.tag1\ p_j\ t_i.tag2$" as a phrase
    $hits2 \leftarrow$ # of hits of querying "$t_i.tag2\ p_j\ t_i.tag1$" as a phrase
    $ratio_j.F \leftarrow \frac{hits1}{hits2}$
    $ratio_j.R \leftarrow \frac{hits2}{hits1}$
  end for
  $maximum_F \leftarrow \max(ratio_j.F)$ over all j
  $maximum_R \leftarrow \max(ratio_j.R)$ over all j
  $maximum \leftarrow \max(maximum_F, maximum_R)$
  if $((maximum = maximum_F)$ and $(maximum_F \geq threshold))$ **then**
    $I \leftarrow I \cup\{\langle$tag1, tag2, IS-A$\rangle\}$
  else
    if $((maximum = maximum_R)$ and $(maximum_R \geq threshold))$ **then**
      $I \leftarrow I \cup\{\langle$tag2, tag1, IS-A$\rangle\}$
    end if
  end if
end for
Return I

relationships: (1) Pattern 1: NP_1 such as NP_2; (2) Pattern 2: NP_1 including NP_2; (3) Pattern 3: NP_1 especially NP_2.

Our HAS-A relationships are supersets of meronymy (part-of relationships), and are not limited to the physical perspective. We consider two noun phrases NP_1 and NP_2 to have a HAS-A relationship (with NP_1 as the parent) if one of the following statements is true: (1) NP_2 is a part of NP_1. E.g., "body" is a part of "human"; or (2) NP_1 has/have NP_2. E.g., "human" has "mind" and "google" has "googleMaps"; or (3) NP_1 may have NP_2. E.g., "human" may have "disease".

From the existing lexico-syntactic patterns mentioned in the literature such as [**?**,**?**], we use three following patterns to detect HAS-A relationships:(1) Pattern 1: NP_1's NP_2; (2) Pattern 2: NP_2 of the NP_1; (3) Pattern 3: NP_2 of NP_1.

While patterns 1 and 2 are among the most common English patterns [**?**], pattern 3 is not. However, pattern 3 can be used to detect HAS-A relationship between tags such as the tuple ⟨Coffee, Caffeine⟩.

All patterns for a relationship are fed into a search engine. If the largest ratio of a pattern is above a threshold, that tuple is labeled with the corresponding relationship and added to the ontology. Algorithm 5 shows the IS-A detection algorithm. The HAS-A algorithm is similar, but requires that pattern 1 *and* one of patterns 2 and 3 are above the threshold. Both thresholds were found experimentally. In our experiments, the IS-A threshold was 7 and HAS-A threshold ranged from 20 to 50.

6 Exploiting Co-parents to Find More IS-A Relationships

Examining the ontology built thus far reveals an interesting property when pairs of tags share the same child. Consider the following example: the ontology may contain "fiction → urban-fantasy" and "fantasy → urban-fantasy", where "fiction" and "fantasy" are both parents for "urban-fantasy" w.r.t. the IS-A relationship.[5] (Figure 1). However, the IS-A relationship between "fiction" and "fantasy" may be missing. One possible reason for this is that people tend to use the more specific tags leading to "fiction → urban-fantasy" and "fantasy → urban-fantasy", so that "fiction → fantasy" does not occur above the relatively high threshold needed to avoid noise.

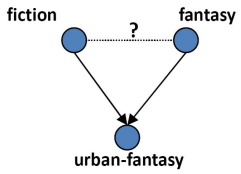


Fig. 1. Concept of co-parents in an ontology

Hence we have the following hypothesis: in a co-parent structure (e.g., Figure 1) it is more likely than usual that the two parents are in an IS-A relationship. Hence, we include the following additional step (Algorithm 6) to ONTECTAS: for such co-parent pairs, we re-examine the pair's confidences under a lower threshold and extract candidate tuples for an IS-A relationship.

As a final step of the ONTECTAS algorithm, following standard practice in ontology extraction algorithms, if the graph of relationships is disconnected, we add a generic "Entity" root node and make it the parent of all orphan nodes.

7 Experiments

7.1 Datasets and Assumptions

Our experiments used four real datasets: 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 1 shows the characteristics of the datasets. User information is not available in the IMDb dataset, so competing algorithms were unable to create ontologies from it.

To show that general purpose ontologies are insufficient, we validated that WordNet misses many relationships between terms *even when it contains both terms*. To show this, we evaluated a sample ontology (from Del.icio.us) both

[5] Here, —fiction" → "urban-fantasy" means "urban-fantasy" IS-A "fiction".

Algorithm 6 Co_Parent_Pruning

Input: (T) A set of tuples with IS-A relationships in form $\langle parentTag, childTag \rangle$; ($F$) A set of frequent itemsets
Output: (T') An enhanced set of tuples with IS-A relationships
1: $T' \leftarrow T$
2: $G \leftarrow$ A graph where each tuple in T corresponds to an edge from $parentTag$ to $childTag$.
3: $S \leftarrow$ All tuples of tags $\langle parent_1, parent_2, child \rangle$
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** $\langle parent_1, parent_2, child \rangle \in S$ **do**
5: **if** $\{parent_1, parent_2\}$ is frequent and if it satisfies lower forward and reverse confidence thresholds **then**
6: Add $\langle parent_1, parent_2 \rangle$ to T' with the more frequent tag as the parent.
7: **end if**
8: **end for**
9: Return T'

Table 1. Corpus Details in Some Collaborative Tagging Systems

	Del.icio.us (Dec. 2007)	CiteULike (Jan. 2010)	IMDb (Nov. 2009)	LibraryThing (corpus from Delft*)
Number of Tags	6,933,179	431,160	2,593,747	10,469
Number of Items	54,401,067	2,081,799	356,162	37,232
Number of Users	978,979	60,220	N/A	7,279
Number of Tag Assignments	450,113,886	7,922,454	2,625,237	2,415,517

* http://homepage.tudelft.nl/5q88p/LT

manually and by using all parent-child senses (meanings) 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 ONTECTAS is in the pool of parents for that child, that relationship is labeled as correct. We limited our experiments to relationships where both parent and child term exist in WordNet. This gives WordNet an advantage since many tags do not appear in WordNet at all. In this case, we found WordNet is missing 26.9% of manually validated relationships discovered by ONTECTAS. For example, WordNet contains 3 senses for "python", but none of these senses is related to programming; as a result, "programming → python" is missing in WordNet.

Table 2 shows the results of validating all the relationships extracted by ONTECTAS on the Del.icio.us dataset. Next, we considered only relationships detected by ONTECTAS where both terms in the relationship existed in WordNet. Table 3 shows that there are many relationships between terms in WordNet that WordNet will not find.

Using Tables 2 and 3, the percentage of missing relationships in WordNet is calculated in Table 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 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.

Table 2. Precision for all relationships

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

Table 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

Table 4. Percentage of missing relationships in WordNet

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

7.2 Validating Assumption for User's Tagging Behavior

ONTECTAS is built on the hypothesis that tags assigned by a group of users to tag the same item tend to contain both a term and its parent. 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

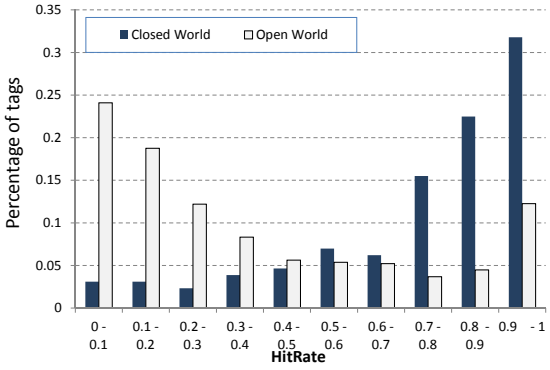


Fig. 2. Histogram of HitRate values

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).

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 2 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 2 validates that people use parent and child tags together. In the closed world, more than 83% 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 24% 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-Arelationship discovered by WordNet. It is worth noting two factors in this open world that strengthen our hypothesis: (1) more than 75% of tags occur with some parent and (2) more than 31% of child tags have a hit rate larger than 0.5. This is quite high, which validates our hypothesis that a group of people tag items with both a tag and its parent, and explains why our recall is so high (as will be shown in Section 7.5).

7.3 Evaluation of ONTECTAS in Detecting HAS-A Relationships

Table 5 shows the precision of ONTECTAS in detecting HAS-A relationships. *None of the other competing algorithms address* HAS-A *relationships from CTSs.*

Table 5 only reports precision for ONTECTAS, the first algorithm to detect HAS-A from CTS data.

One challenge in detecting HAS-A relationships was that pattern-based search engine queries such as "human's middle" and "middle of human" are frequently part of phrases such as "human's middle finger" and "middle of human history". Clearly, there is room for improvement in ONTECTAS' precision in HAS-A detection, which we plan to address in future work.

Table 5. Precision in detecting HAS-A relationships

	Del.icio.us	CiteULike	LibraryThing	IMDb
Precision	51.6%	61.9%	55.5%	33.3%

7.4 Evaluation of ONTECTAS in Detecting IS-A Relationships

In the following, we focus on IS-A relationships. All competing algorithms do not distinguish between IS-A and other relationships such as synonyms, whereas we clearly isolate IS-A relationships. We lump all other relationships into ANY and compare the performance of ONTECTAS on IS-A with that of other algorithms on IS-A and ANY, giving them an advantage, since in this evaluation, we do not give credit to ONTECTAS for correctly finding HAS-A relationships. We use the following standard performance measures: (1) Precision: We consider the precision of ONTECTAS on IS-A with that of other algorithms on IS-A+ ANY. Precision for both is the number of correct edges over the number of all edges. (2) Maximum depth and average depth of the IS-A taxonomy. (3) Average number of children. A higher value of the last two measures implies richer ontology is extracted. In addition, following [**?**], we compare all algorithms with a gold standard to see how they fare in trying to recreate manually-curated ontologies.

For depth and breadth metrics, we calculate these metrics on an ontology with only correct relationships to ensure algorithms cannot earn an artificially and unfairly high score on these by finding many incorrect relationships!

Absolute recall for ontology extraction from a large CTS is very hard to measure. Instead, we 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. Relative recall for an algorithm is the number of valid IS-A relationships found by the algorithm divided by the total number of valid IS-A relationships found by all algorithms.

7.5 Comparing ONTECTAS to Other Algorithms

We compare ONTECTAS with the four algorithms from Section 2: 1) the algorithm from [**?**] (abbreviated "LFZ") 2) the DAG algorithm [**?**] ("DAG-ALG") 3) Schmitz's algorithm [**?**] ("Schmitz"), and 4) Barla and Bieliková's algorithm [**?**] ("BB"). Since these algorithms cannot process the IMDb dataset due to the

lack of user information, we only compare them on Del.icio.us, LibraryThing, and CiteULike.

To have a fair comparison, we implemented the above algorithms as closely as possible to the way their authors had implemented them; 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.

Validating the edges manually required that each algorithm output a small number of edges. To do so, we put another threshold on the number of times a tag, an item, or a user must occur in order to be considered. To be fair, we used the same threshold to ensure that each algorithm output fewer than 150 edges.

Figure 3(a) shows the algorithms' precision for both IS-A relationships (the lower bars) and ANY relationships (the higher bars); for IS-A relationships, the precision of ONTECTAS is 0.50 for Del.icio.us, 0.48 for LibraryThing and 0.29 for CiteULike. ONTECTAS outperforms the precision of all other algorithms on all datasets. We also compare our precision on IS-A with that on IS-A+ ANY for the other algoritms since they do not distinguish IS-A from non-IS-A. Even then ONTECTAS outperforms the other algorithms in del.icio.us and CiteULike. On LibraryThing, the performance is close to the winner.

Figure 3(b) compares the algorithms' relative recall for IS-A relationships. ONTECTAS is the best performer for all three datasets. One reason for DAG-ALG's bad relative recall is that it detected many popular tags such as "web" and "software" as subjective tags, and pruned them before discovering the edges. BB had relatively low precision and recall in CiteULike because it detected many relationships with the tag "no-tag", which is a popular tag rather than an ontological tag. Figure 3(e) shows the relative recall for any relationships. As we can see, ONTECTAS is still the best performer, though by a smaller margin.

Figures 3(c), 3(d), and 3(f) measure the depth of the validated ontology detected by each algorithm for both IS-A (lower bars) and ANY relationships (higher bars). These measures quantify the richness of the ontology. If there are multiple paths from the root to a node n, the depth is the longest path. Because the other algorithms find just ANY relationship between elements in an ontology, rather determining the types of relationships, like ONTECTAS does, we measure both the IS-A relationships and ANY relationships found. We do not consider HAS-A since no other algorithms detect it. Notice that this gives an advantage to the competing algorithms. For the depth metrics, other algorithms usually find a long chain with combination of synonyms and IS-A relationships. Since ONTECTAS detects mostly IS-A or HAS-A (and not synonyms), maximum depth for ANY relationship in ONTECTAS is close to maximum depth of IS-A relationship because in general of chains containing IS-A and HAS-A are rare.

For IS-A relationships, ONTECTAS has the highest maximum depth for two out of three datasets. For the average number of children, ONTECTAS has the best performance for CiteULike, is roughly tied for Library thing, and is second best for Del.icio.us.

The results for average number of children for IS-A are similar to those for average number depth. Even when competing algorithms are given credit for

ANY relationships and ONTECTAS only for finding IS-A, ONTECTAS performs fairly well. This is because there are so many IS-A relationships detected as compared to the other relationship types.

For all of the depth/children metrics, we note that *all algorithms perform markedly better using our removing verb phrases preprocessing step*. This step removed many non-ontological tags such as "to-read" in the Del.icio.us dataset. By applying this to all algorithms, we have improved all algorithms' performance, not just ONTECTAS's. Figure 3 also shows that most of the algorithms performed better on most measures for the Deli.icio.us and LibraryThing datasets than on CiteULike. This validates that the tags in these datasets are of better quality than the ones in CiteULike. This shows that we can compare different CTSs on the quality of tagging actions, using an ontology creation algorithm.

In summary, ONTECTAS outperforms the four other algorithms on precision and relative recall for IS-A relationships, and does well on the structural metrics of maximum depth, average depth and average number of children.

7.6 Comparing with a Gold Standard

Following [**?**], we compared how the algorithms extracted IS-A relationships against a "gold standard" ontology — the concept hierarchy from the Open Directory Project (ODP) [6]. To judge precision, recall, and F-measure, we use the lexical and taxonomic metrics from [**?**]. The lexical metrics measure how well the algorithms did in recreating the *concepts*, and the taxonomic metrics show how well the algorithms did in recreating the *structure*. Notice that comparing with a static ontology considered as gold standard has its problems since it may miss important concepts and relationships and a good algorithm that finds concepts and relationships manually verified to be correct may get penalized unfairly. We will return to this point.

Formally, given an output ontology $O_R = \{C_R, root, \preceq_{C_R}\}$ (where C is a set of concepts, *root* is the root and $\preceq_{C_R}$ is a hierarchy), and a gold standard $O_G = \{C_G, root, \preceq_{C_G}\}$, the lexical precision is defined as

$$LP(O_R, O_G) = \frac{|O_R \cap O_G|}{|O_R|} \tag{1}$$

the lexical recall is defined as

$$LR(O_R, O_G) = \frac{|O_R \cap O_G|}{|O_G|} \tag{2}$$

The lexical F-measure is defined as:

$$LF(O_R, O_G) = \frac{2 \cdot LP(O_R, O_G) \cdot LR(O_R, O_G)}{LP(O_R, O_G) + LR(O_R, O_G)} \tag{3}$$

[6] http://dmoz.org

The taxonomic metrics are based on the common semantic cotopy (i.e., ancestors and decedents of a node) of each concept in both ontologies [**?**]. Formally, given two ontologies O_1 and O_2, the common semantic cotopy of a concept c is

$$csc(c, O_1, O_2) = \{c\prime | c\prime \in C_1 \cap C_2 \wedge (c\prime \prec_{C_1} c \vee c \prec_{C_1} c\prime)\} \tag{4}$$

Given a resulting ontology O_R and a gold standard O_G, the local taxonomic precision of the concept c is defined as

$$tp(c, O_R, O_G) = \frac{|csc(c, O_R, O_G) \cap csc(c, O_G, O_R)|}{|csc(c, O_R, O_G)|} \tag{5}$$

Thus, taxonomic precision is defined as

$$TP(O_R, O_G) = \frac{1}{|C_R \cap C_G|} \sum_{c \in C_R \cap C_G} tp(c, O_R, O_G) \tag{6}$$

Taxonomic recall is defined as

$$TR(O_R, O_G) = \frac{1}{|C_R \cap C_G|} \sum_{c \in C_R \cap C_G} tp(c, O_G, O_R) \tag{7}$$

Taxonomic f-measure is defined as

$$TF(O_R, O_G) = \frac{2 \cdot TP(O_R, O_G) \cdot TR(O_R, O_G)}{TP(O_R, O_G) + TR(O_R, O_G)} \tag{8}$$

Table 6. Gold standard based lexical metric results for ontologies produced by five algorithms. Subtrees associated with manually selected root concepts are extracted and compared with the corresponding part in the gold standard.

Root	Lexical Precision(LP)					Lexical Recall(LR)					Lexical F-measure(LF)				
	ONT	LFZ	BB	DAG	Schmitz	ONT	LFZ	BB	DAG	Schmitz	ONT	LFZ	BB	DAG	Schmitz
software	0.2632	**1**	0.3333	0.5	0.2174	**0.0481**	0.0081	0.0153	0.0434	0.0106	**0.0813**	0.0161	0.0293	0.0799	0.0202
programming	0.5625	**1**	0.4375	0.375	0.1819	0.0104	0.0006	0.0175	**0.0272**	0.0039	0.0204	0.0012	0.0337	**0.0507**	0.0076
web	0.0789	**0.5**	0.0313	0.0741	0.0357	**0.1163**	0.0233	**0.1163**	0.0698	0.0233	**0.0940**	0.0445	0.0493	0.0719	0.0282
technology	0.0653	**0.5**	0.0907	0.3333	0.2	0.0238	0.001	**0.0859**	0.001	0.001	0.0349	0.0020	**0.0882**	0.0020	0.0020
photography	0.25	**0.5**	0.75	0.0625	0.0556	**0.009**	0.0023	**0.009**	0.0023	0.0023	0.0174	0.0046	**0.0178**	0.0044	0.0044
art	0.25	**1**	0.2222	**1**	0.05	**0.4444**	0.1111	0.2222	0.1111	0.1111	**0.3200**	0.2000	0.2222	0.2000	0.0690
6 largest	0.256	**0.6667**	0.1511	0.6129	0.125	**0.0308**	0.0045	0.0293	0.028	0.0065	**0.0550**	0.0089	0.0491	0.0536	0.0124
Overall	0.261	0.743	0.183	**0.745**	0.128	0.240	0.006	0.244	**0.025**	0.007	0.044	0.011	0.043	**0.049**	0.014

The resulting ontologies are still fairly small comparing with the ODP ontology, which has 763,378 concepts overall; only 3,946 concepts were found by any of the algorithms. We looked at the 25 highest-level concepts common across the five algorithms.

Table 6 shows the lexical metrics for all five algorithms to see how similar the output *concepts* are to the gold standard. Table 7 shows the taxonomic

Table 7. Gold standard based taxonomic metric results for ontologies produced by five algorithms. Subtrees associated with manually selected root concepts are extracted and compared with the corresponding part in the gold standard.

Root	Taxonomic Precision(TP)					Taxonomic Recall(TR)					Taxonomic F-measure(TF)				
	ONT	LFZ	BB	DAG	Schmitz	ONT	LFZ	BB	DAG	Schmitz	ONT	LFZ	BB	DAG	Schmitz
software	0.7102	0	**1**	0.2739	**1**	0.7188	0	0.4857	**0.762**	0.25	**0.7144**	—	0.6538	0.4029	0.4
programming	0.8889	0	0.9405	0.5333	**1**	**1**	0	0.69	0.95	**1**	0.9411	—	0.7960	0.6831	**1**
web	**1**	0	0.6667	**1**	0	**1**	0	**1**	**1**	0	**1**	—	0.8000	**1**	—
technology	0.5513	0	**0.6644**	0	0	**0.9565**	0	0.9031	0	0	0.6994	—	**0.7655**	—	—
photography	**1**	0	**1**	0	0	**1**	0	**1**	0	0	**1**	—	**1**	—	—
art	**1**	0	**1**	0	0	**1**	0	**1**	0	0	**1**	—	**1**	—	—
6 largest	0.5956	0	0.5341	0.2348	**0.7**	0.7983	0	0.7494	**0.8041**	0.35	**0.6822**	—	0.6236	0.3635	0.4667
Overall	**0.480**	0.077	0.434	0.123	0.329	0.723	0.023	0.711	**0.783**	0.256	**0.577**	0.035	0.539	0.212	0.288

metrics, which try to capture how similar the output *structure* is to the gold standard. Bolded entries represent the best performance. Because the 25 highest level common concepts were very uneven in size, we only performed a detailed analysis of the 6 largest subtrees — otherwise algorithms would be testing against subtrees that were only one or two concepts large.

ONTECTAS has the second highest overall lexical recall and f-measure, which shows that it finds the desired concepts well. While DAG had the highest lexical precision and f-measure, and BB had the highest lexical recall, they both did very poorly on taxonomic precision, leading to a low taxonomic f-measure.

LFZ had a very good lexical precision; however, this is achieved by reporting a very small number of correct concepts. ONTECTAS is superior to LFZ in terms of all three taxonomic measures. Unlike for lexical recall, there are some cases where the taxonomic recall is zero. This means there is no validated hierarchy corresponding to this concept in the output produced by that algorithm. This often occurs when single nodes rather than their surrounding nodes are matched with the gold ontology. A very flat taxonomy can also suffer from this when leaf nodes are matched.

As we show in Tables 6 and 7, when we considered only the 6 largest subtrees, ONTECTAS had the best lexical and taxonomic f-measure.

Algorithms with zero recall cannot have their f-measure calculated, so their f-measure is indicated by a dash. In this experiment LFZ had a very shallow taxonomy, and its taxonomic recall and f-measure suffer accordingly.

Comparing to a gold standard shows how well algorithms do against a manually created ontology. But since a gold standard ontology is static, this metric may unfairly penalize algorithms that genuinely find correct concepts and relationships. E.g., "dialect" and "software IS-A technology" is incorrect according to this standard. Thus, comparing algorithms should take into account other components discussed above as well.

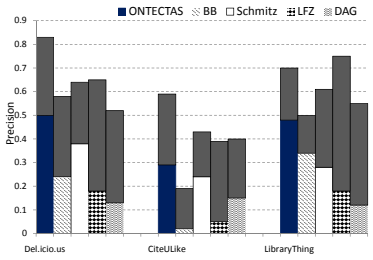
8 Conclusion and Future Work

We proposed an algorithm (ONTECTAS) for building ontologies of keywords from collaborative tagging systems. ONTECTAS uses association rule mining, bi-gram pruning, exploiting pairs of tags with the same child, and lexico-syntactic

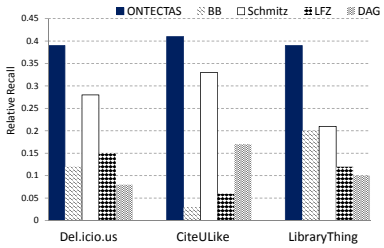
patterns to detect relationships between tags. We also provided a thorough analysis of ONTECTAS and how it compares to other algorithms. Some of the important open problems include detecting spam users, improving accuracy of ontology extraction via supervised learning and by means of incorporation of part-of-speech detection. Our ongoing work addresses some of these.

Acknowledgements

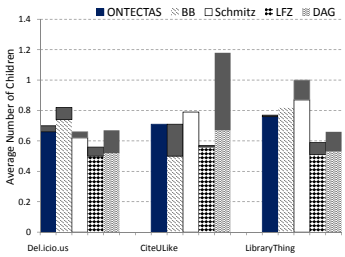
This research is funded by a grant from NSERC Canada.



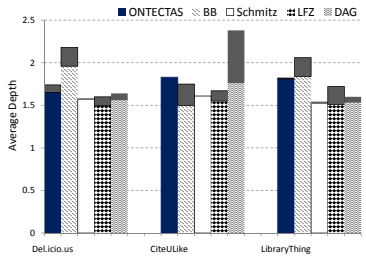
(a) Precision



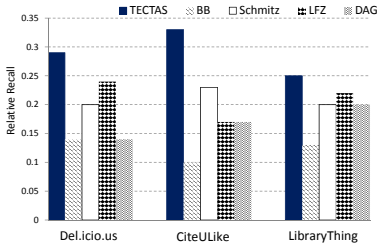
(b) Relative Recall — IS-A



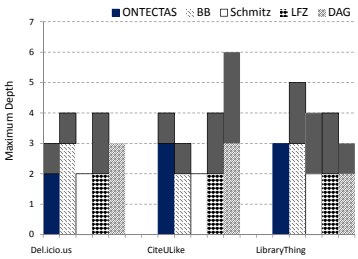
(c) Average Number of Children



(d) Average Depth



(e) Relative Recall (any relationship)



(f) Maximum Depth

Fig. 3. Comparison of ONTECTAS to other algorithms for different metrics. Lower bars show IS-Arelationships and higher bars show "any" relationships.