TAGSENSE: MARRYING FOLKSONOMY AND ONTOLOGY

by

ZIXIN WU

(Under the Direction of Amit P. Sheth)

ABSTRACT

Tagging communities is a featured Web 2.0 phenomenon, where users describe a Web resource by using keywords (called tags). This behavior can be viewed as cooperative meta-data extraction and annotation. While these tagging communities become more and more popular, their information retrieval mechanism is still keyword based, thus it is difficult to achieve high precision and recall rates because of word ambiguity and lack of semantics. In this thesis, we combine the approaches of folksonomy and ontology to improve recall rate and ranking of query results. We index Web resources by the meanings of tags instead of strings, and we provide context-aware multi-ontologies semantic search capability which utilizes relationships in ontologies. The evaluations performed on a subset of Flickr's photos indicate that our users spent significantly less time and effort in finding what they want on our system than on Google Desktop on the same datasets.

INDEX WORDS: Folksonomy, Ontology, Semantic Web, Semantic Query, Information Retrieval, Precision, Recall, Synonym, Polysemy, TF-IDF, Context

TAGSENSE: MARRYING FOLKSONOMY AND ONTOLOGY

by

ZIXIN WU

B.E., Beijing Broadcasting Institute, China, 2002

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2007

TAGSENSE: MARRYING FOLKSONOMY AND ONTOLOGY

by

ZIXIN WU

Major Professor: Amit P. Sheth

Committee: John A. Miller
Prashant Doshi

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2007

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Amit P. Sheth for his support and great encouragement to research. Also thanks to Dr. John A. Miller and Dr. Prashant Doshi for offering their experience and expert advice. My appreciation also goes to Kunal Verma for guiding me to the research area.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

Coined by O'Reilly Media in 2004, the term Web 2.0 [1] refers to the second-generation of the Web, whose significant differences from the “standard” Web 1.0 are users' social behaviors on the Web, specifically their contribution to the content, and the importance of data and Web services.

Recently tagging communities as one of the featured Web 2.0 phenomena are more and more popular. The users comprising these communities are free to choose whatever keywords (called tags) they like to describe a Web resource, such as a bookmark in Del.icio.us [2], a photo in Flickr [3], a video clip in YouTube [4], or an academic paper in CiteULike [5]. The behavior of massive tagging in social context and its product - tags for Web resources, are called folksonomy [6-8]. The users may issue queries to retrieve the Web resources whose tags match the keywords in the queries. This approach is easy to use, requires no training or expertise, thus is welcome by large number of users.

At a first glance, folksonomy seems trivial and nothing new. People have been using keywords to describe items for many years, such as in academic papers. And keyword based information retrieval approaches prevail these days, such as in Google [9] and Yahoo [10] search engines.

However, a close look may reveal the value of folksonomy. It is an approach of metadata extraction from the needs of users. Most search engines have a crawler to get documents from the Web and identify keywords. They are usually domain and user independent. In folksonomy, however, users may put tags that are only interesting to them. Folksonomy provides a more targeted set of tags that would be used by the users with similar interests. For instance, an environmentalist may only concern the air pollution of a nuclear test, not the expense of the test. Folksonomy also makes metadata extraction from multimedia Web resources much easier: the users see a picture or a video and describe what it is about. Folksonomy is different from the traditional ways where people use keywords: it is produced by many users in social context. The tags may not be in canonical forms as in a controlled vocabulary. For instance, authors of academic papers usually have implicit agreement on terms: no author writes “Info retrieval” for “Information Retrieval”. Finally, not only the information producers can put tags, but all other users, such as information consumers, may also put tags. Thus the users may use their tags to vote what is important for a Web resource.

While tagging provides a convenient way to annotate Web resources, all the current tagging systems (except some systems that provide geographical annotations) rely on keyword-based search techniques, and are hard to achieve high precision and recall rates. The fundamental reasons are (1) tags may be ambiguous, and (2) there is no explicit relationship between tags, so often times it is hard to filter out the wanted information.

This lets us think of a candidate to compensate for these drawbacks. Ontology [11], the key enabler of the Semantic Web [12], is a quite different mechanism from folksonomy. Ontologies

are designed for enabling knowledge sharing and reuse. They describe the entities, their attributes, and relationships between them in an explicit way, so that software agents can communicate about a domain of discourse if they commit to a common ontology [13].

Ontologies state knowledge in an unambiguous way by using URIs [14] for its entities and their relationships. The relationships are stated explicitly, so that a software agent can utilize them to filter out information. For example, we may find out all the sons of a person stated in an ontology. Ontologies also facilitate reasoning, for example, systems may easily infer a father of a father of person *p* is an ancestor of *p*, due to the relationships stated explicitly in an ontology.

A question that has seen much interest from the research community is that what Semantic Web technologies can do for folksonomy and vice versa? Actually we can see some trends in using structured data in folksonomies these days. For instance, the users in Flickr [3] may "geotag" their photos by dragging and dropping them on a map to annotate the locations where the photos were taken. Panoramio [15] has similar feature but their system automatically detects geography terms in the input tags, such as city names, and asks users to choose the correct location if a tag is ambiguous. Flickr also provides the concepts and tools of Sets and Collections [16], and SmartSetr [17] for the users to organize their photos into a hierarchical structure.

We want to utilize ontologies in general cases, i.e. not restricting to a specific domain, such as geography. We propose an approach in this thesis which takes the advantages from both folksonomy and ontology. First, we do not add any burden to our users: they should be able to use only keywords/tags to describe and search Web resources. Second, we do not expect our

users have Semantic Web background. Third, we want to utilize ontologies as background knowledge in information retrieval.

In most cases, the reasons why a user has to change their queries multiple times for a goal is because either (a) they cannot get what they want due to low recall rate, which is most likely caused by synonyms and/or lack of inference capability in the systems, or (b) they get too many records that they do not want due to low precision rate. Our goal is to shorten the time and effort that a user has to spend in order to retrieve wanted information. Specifically, (1) improve recall rates by considering synonyms and enabling semantic search (2) improve result ranking by putting the most appropriate items on the top of the query results.

Our approach is to index Web resources by word senses (meanings) instead of keywords, and then match the senses to ontological concepts (either classes or instances). We provide semantic search which has basic inference capability and filters out information based on the relationships defined in ontologies, and we propose the Most-Desired Sense Ranking approach to rank the senses that matches a query. The Most-Desired Senses Ranking approach is to show one record (photo in our case) for each sense that matches the query and let the user choose the best one to him/her, and then the system rank the other matched senses based on the user's selection. The evaluations indicate that our users spent significantly less time and effort in finding what they want on our system compared with Google Desktop for the same datasets.

This thesis is organized as follows. Chapter 2 introduces folksonomy and ontology, and Chapter 3 presents related work. In Chapter 4, we give an overview of our approach, and details in Chapter 5 to 8. We describe how to clean up our datasets from Flickr in Chapter 5, how to

index Web resources by senses in Chapter 6, how to utilize ontologies in Chapter 7, and how to perform semantic searches and rank ontologies in Chapter 8. Chapter 9 presents our datasets, use cases, and evaluations. We conclude in Chapter 10 with a look at future work.

CHAPTER 2

BACKGROUND

1. Folksonomy

Most Web 2.0 [1] websites provide an easy-to-use way for their users to describe Web resources they found or published. The users are free to use any words (called tags) to describe a Web resource, which can be a Web page (such as in Del.icio.us [2]), a photo (such as in Flickr [3]), a video clip (such as in YouTube [4]), etc. The behavior of massive tagging in social context and its result - tags for Web resources by users, are called folksonomy [6-8]. The users may issue queries to retrieve the Web resources whose tags match the keywords in the queries.

We may wonder for what incentives the users put tags to describe Web resources. Marlow et. al. discuss some of them in [18], such as for future retrieval (such as bookmarks in Del.icio.us), contribution and sharing (such as uploading travel photos in Flickr), attracting attention, play and competition, self presentation, and opinion expression.

Tagging is easy to use, requires no training or expertise, thus is welcome by a large number of users. One of the advantages of folksonomy is that it does not restrict the available terms that the users can choose, and there is no mandatory taxonomy the users have to conform to. To some extent, folksonomy can be viewed as multi-dimensional indexing.

At a first glance some people may consider folksonomy trivial: people have been using keywords to describe objects for many years, and keyword-based search engines prevail these

days. Yet there are much more behind this simple approach. First, it is an approach of metadata extraction from the needs of users. Users may put only the tags which describe what is interesting to them, for example, putting tags about a house in a photo but not the dog in it, thus ignore noise to the taggers. Second, popular tags prevail and tags for a Web resource converge over time [19] in a community. This makes information retrieval easier, since an informal controlled vocabulary is self-emergent (yet synonyms still exist). Third, it makes metadata extraction from multimedia Web resources easier: it is hard to create a program to recognize a flower in a photo, but the users are willing to put tags such as “flower” to the photo.

Although keyword tagging is nothing new, when it happens in the social context and by a large number of users with similar interests or purposes, the folksonomy triad (the person tagging; the Web resource being tagged; and the tags being used on that object) [20, 21] become powerful. We can use two of the elements to find a third element. For example, we may find persons with similar interests by comparing the Web resources they tagged and the tags they used. One of the most important features of folksonomies is power law distribution of tags [19]. That is, for a particular Web resource in broad folksonomies, there are a small number of tags that are popular (repeated heavily). Broad folksonomies [22] (like in Del.icio.us) is where any user may put tags for a Web resource, and contrastively, narrow folksonomy is where only the owner of the Web resource can put tags.

While tagging is a convenient way to annotate Web resources, it is generally hard to achieve high precision and recall rates, mainly because: (1) tags may be ambiguous, and (2) there is no explicit relationship between tags, so often times it is hard to filter out the wanted information.

An example is that we want to see photos of a bird “turkey” in Flickr. The majority of the photos returned by the query “turkey” are about the country Turkey in Europe. In order to get higher precision, we may add a keyword such as “bird” in the query. While the precision is increased, the recall is decreased: all photos without a tag “bird” will not be shown (or will be shown at the last of the result list).

As another example, we want to see photos about some cities in USA, without specific names in our mind. It is an almost impossible task for current tagging systems, as there is no such knowledge as “Atlanta is a city” in the systems. The users have to try the cities’ names one by one. Note that the fact that the tag “Atlanta” co-occurs quite often with the tag “city” does not imply Atlanta is a city, because other tags such as “street” may also co-occur with the tag “city” quite often, and “Georgia” may also co-occur with “Atlanta” quite often.

2. Ontology

Ontology is an important term in Knowledge Representation and the key enabler of the Semantic Web [12]. A widely accepted definition of ontology is a formal specification of a conceptualization [11]. Conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose [13]. Ontologies are designed for enabling knowledge sharing, reuse, and reasoning. They describe entities, their attributes, and relationships between the entities. Software agents can communicate about a domain of discourse if they commit to a common ontology, that is, their observable actions are consistent with the definitions in the ontology [13]. The explicitness of ontologies also facilitates reasoning.

An ontology language is a formal language used to encode an ontology. OWL [23] is a language for making ontological statements, developed as a follow-on from RDF [24] and RDFS [25], as well as earlier ontology language projects including OIL [26], DAML [27] and DAML+OIL [27]. OWL is intended to be used over the World Wide Web, and all its elements (classes, properties and individuals) are defined as RDF resources, and identified by URIs. OWL currently has three sublanguages (or species): OWL Lite, OWL DL, and OWL Full. Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded.

Ontologies are often domain specific. They represent concepts in very specific ways, and due to different perceptions of the domain based on cultural background, education, and ideology, different ontologies in the same domain may be incompatible. For example, tips may be defined as part of dinning expenses by an ontology designer in the United States, but an ontology designer in China does not do that, as people in China do not pay tips. In order to exchange information, we may: (1) resolve conflicts in ontology matching / mapping, (2) support multiple ontologies and give weights to them, (3) use upper ontologies[28], such as SKOS [29], OpenCyc [30]. Upper ontologies are large in order to be comprehensive, thus they often take community efforts to build and maintain. They also only represent the perceptions from their creators, thus do not allow incompatible knowledge.

Ontological classification or categorization is the way to organize a set of entities into groups, based on their essences and possible relations [31]. A taxonomy can be seen as a subset of an ontology, which has only class definitions and subsumption relations. As mentioned by [31],

ontological classification works well in the situations where the domain to be organized has small corpus, formal categories, stable entities, restricted entities, or clear edges, and the participants are expert catalogers, authoritative source of judgment, coordinated users, or expert users.

From the above observations, ontology or taxonomy alone is not a good candidate for knowledge or information exchange in social contexts, where the users come from various disciplines with different backgrounds, the set of users involved change over the time, and the knowledge in social contexts changes relatively more frequently. These features indicate that it is hard to achieve a commitment among all users, and it is hard for the categorizers to guess what their users are thinking when they create the categories, and make predictions about the changes of the categories in the future.

3. Comparison

It seems that folksonomy and ontology are different mechanism with contrary styles. Table 1 gives a comparison of their advantages and disadvantages. “(+)” represents advantage, and “(-)” represents disadvantage.

Table 1: Comparison of Folksonomy and Ontology

	Ontology	**Folksonomy**
Flexibility	(-) Taxonomies limit the dimensions along which one can make distinctions [8]	(+) Massively dimensional (one dimension per potential term) [8]
	(-) Disallows inconsistency (but this makes reasoning easier)	(+) Allows inconsistency
	(-) Usually restricts to one domain. Exceptions are upper ontologies.	(+) Across domains, disciplines, cultures, and languages
	(-) Relatively hard to build, maintain, or change	(+) Self-evolves or emergent
Usability	(-) High entry cost: often needs experts	(+) Low entry cost: no training necessary
Accuracy	(+) Concepts and relationships are unambiguous	(-) Tags may be ambiguous
Expressiveness	(+) Explicit relationships between concepts	(-) Implicit relationships between tags
Reasoning	(+) Facilitates reasoning	(-) Very hard to reason, if not impossible

The following is a line-by-line explanation of table 1.

(1) A concept often has multiple properties. For example, an apple is a fruit, juicy, sweet, and contains lots of vitamins. When building a taxonomy, we need to determine which attribute is the most important, and then the second, the third, etc, and consequently determine the categorization tree. Therefore, a taxonomy may work well in a particular domain, but may not in all domains. A fruit juice manufacturer concerns if an apple can produce lots of juice, while a nutritionist concerns how many types of vitamins an apple has. So their ideal categorization trees are different. On the contrary, folksonomy neither has restriction on the number of dimensions, nor gives priority to any dimension.

(2) Consistency is an important feature of ontologies or taxonomies. It is a prerequisite of ontological commitment, and also makes reasoning easier. If the domain experts determine dolphin is a mammal, every user has to agree on this if he / she needs to use this particular part of the taxonomy. Folksonomy is more democratic: one may tag “dolphin, mammal”, while another one may tag “dolphin, fish”.

(3) Ontologies are often domain specific, due to the commitments, effort, time, and expertise requirements in building and maintenance. There are some upper ontologies, such as [29, 30]. They are large in order to be comprehensive, but consequently hard to build and maintain. Folksonomies are not domain specific at the first place: a knowledge management consultant can find valuable information from an information architect because an object is tagged by both communities using their own differing terms of practice [22].

(4) Ontology maintenance and evolution is an active research topic, which usually involves much effort and expertise. Folksonomy evolves itself automatically: whenever a new slang becomes popular, more and more users tag with it.

(5) Building and using ontologies usually need domain experts or collaborations of users with domain knowledge and field knowledge of ontology, such as RDF [24] and OWL [23]. Folksonomy has an obvious advantage at this point: everything is keyword(s), everyone can tag as long as he / she is able to type.

(6) One of the most important features of Semantic Web is unambiguity of terms. This is exactly the most important shortcoming of natural languages (thus folksonomy) in information retrieval. For example, the keyword “apple” may refer to the fruit apple, or the company Apple (assuming case insensitive).

(7) Ontologies usually define explicit relationships between concepts. This feature makes it much easier to select the wanted information. Contrastively, folksonomies do not have any explicit relationships between tags, due to the convinience of input. For example, a photo with three tags “apple”, “banana”, and “green” cannot tell if the apple is green or the banana is green. So systems cannot precisely select photos of a green banana.

(8) Unambiguity, explicitness, consistency make reasoning on ontology easier. For example, we may define a rule that A and B are colleagues if they work for the same company, and then we may infer John and Mary are colleagues if the ontology (or ontologies) states John work for company X, and Mary also work for company X. In folksonomy, however, one has to write sophisticated programs to do reasoning, and usually not in a scalable way.

The above comparison indicates that folksonomy and ontology are complimentary approaches to information exchange. This also suggests that combining folksonomies and ontologies may alleviate their shortcomings and provides a new mechanism which excels either of them alone.

CHAPTER 3

RELATED WORK

There are lots of discussion and research projects going on about folksonomy and combining ontology and folksonomy. We may categorize some important work related to this thesis into the following areas.

Folksonomy

Marlow et al. have an early paper [18] describing folksonomy, such as the model of tagging systems, user incentives, and patterns of the tags in Flickr [3].

Shirky compares ontological classification with folksonomy in an article [31]. He states that ontological classification does not work well in the domains without formal categories, clear edges, with unstable or unrestricted entities, and most users are uncoordinated users or amateur users. He suggests that folksonomy is a flexible way which performs well in these areas.

Halpin et al. [19] analyzed the nature of collaborative tagging systems and discovered that tag distributions converge to power law distributions, and the tags used for annotating a Web resource converge over time.

Semantics for social networks

Discovering and utilizing semantics for social networks attracts more and more attentions from the researchers in the Semantic Web community.

Gruber inspired us by his article [32] where he suggests that Semantic Web technology may benefits the Social Web by providing structured information and prior knowledge, and machines can harvest large amount of human-generated knowledge in the Social Web. In TagCommons project [8, 33], he and other group members proposed to build an ontology to formally describe Tag Data Conceptualization. Some core tagging concepts include Tagging, Tagger, TagSource, TagLabel, TagSpace, etc. There are some interesting discussions in their mailing list. For example, Thomas Vander Wal described his project of searching tags across various tagging services, and suggested the folksonomy triad may be helpful for disambiguation; Nitin Borwankar and Bernard Vatant discussed the idea of “tagging venue” and “tag bundle” that cluster related tags; Tom Heath suggested SPARQL query services over tag data.

Another researcher Mika [21] extends the traditional bipartite model of ontologies to a tripartite model of actors, concepts and instances, and shows how community-based semantics emerge from this model through a process of graph transformation. Other Work on emergent semantics [34] has appeared recently, for example, Schmitz [35] shows how to induce ontology from the Flickr tag vocabulary by using a subsumption-based model; Wu et al. [36] show how emergent semantics can be statistically derived from the social annotations.

Jung et al. [37] propose a three-layered model which involves social network, ontology network and a network of the concepts occurring in the ontologies.

Aleman et al. discovered various semantic associations between the reviewers and authors in a populated ontology to determine a degree of Conflict of Interest. The ontology was created by integrating entities and relationships in FOAF [38] social network and DBLP [39] bibliography.

Sense Disambiguation

Ide et al. surveyed different word sense disambiguation methods in [40], including AI-based methods, Knowledge-based methods, and Corpus-based methods. Wilks et al. [41] combine several simple and independent word sense disambiguation methods and correctly tagged 86% of polysemous words in their test set. Diab et al. [42] present a technique which takes advantage of cross-language lexicalizations of the same concepts.

WordNet [43] is a semantic lexicon for the English language. It groups English words into sets of synonyms called synsets, each of which represents a single lexical concept, organizes them into a conceptual hierarchy like, and includes other links among words according to several semantic relations, such as hyponomy/hyperonymy, antonymy, meronymy, etc. Many sense disambiguation research have been done based on WordNet, for example, Giunchiglia et al. [44] solve the problem of hierarchies or ontologies matching by representing every term in the ontologies as a propositional predicate of WordNet senses. Bouquet et al. [45] present a general methodology for automatically eliciting and representing the intended meaning of schemas in a formal language, which is the result of combining a logical language ALCIO, and IDs of lexical entries in WordNet. Molina et al. [46] use Hidden Markov Models and WordNet for Word Sense Disambiguation. Banerjee et al. [47] discusses the problem of word sense disambiguation in Bangla and its resolution being WordNet like structures. They aimed for a success rate of 92%.

Reference reconciliation is a similar problem, which identifies when different references in a dataset correspond to the same real-world entity. The approach presented in [48] is to use a conclusion of reconciliation for other references and propagate decisions through a dependency

graph. One of their assumptions is that the dataset does not change, which is not the case in folksonomy.

Measuring Similarity

Measuring the similarity of two entities is an essential step in disambiguation. Some well-know methods in Information Retrieval include Jaccard similarity coefficient and Dice's coefficient. One of the drawbacks of these traditional approaches is that they assume different words are totally independent, which is usually not the case in the real world. Ganesan et al. [49] relief this problem by considering the similarity of any two terms in a hierarchical domain structure. There are at least two problems in their approaches: (1) the edge counting method [50] they used to calculate the similarity of two terms in a hierarchy usually does not yield satisfactory result (there is an attempt to improve edge counting result [51]), and (2) their Generalized Vector-Space Model may produce a similarity greater than one if the similarities from the hierarchy are unreasonable, for example, A is 0.9 similar to B, B is 0.9 similar to C, but A is only 0.01 similar to C.

Another type of approach to measuring similarity in a taxonomy is by using Information Content [52]. Resnik [53] presents an approach considering both the structure of a taxonomy, and the informativeness of each term in the taxonomy. He defines the similarity of two concepts as the information content of their most specific common ancestor. Lin [54] normalized the similarity by taking into account the information content of the concepts being considered.

Li et at. [55] combine different approaches nonlinearly, including edge counting, the depth in the hierarchy where the word is found, the density of the subhierachies and the type of link. They claim their measure outperforms existing measures.

Cilibrasi et al. [56] propose another type of measure by utilizing Google hit counts. The approach in [57] considers additional information: the text snippets returned by a Web search engine.

Latent Semantic Analysis

Latent Semantic Analysis (LSA) [58] is a different approach to extract and utilize semantics by statistical computations applied to a large corpus of text. LSA transforms the term-document matrix into a relation between the terms and some concepts, and a relation between those concepts and the documents. Thus the terms and documents are now indirectly related through the concepts. The most significant problem of LSA is that whenever the dataset is changed, the index has to be computed again from the ground up, which is obviously not suitable in social context. In addition, LSA totally rely on the corpus, and cannot utilize existing knowledge, such as existing ontologies.

Semantic Search

Sheth et al. [59] had a commercial product named MediaAnywhere, which is a semantic information retrieval system by the company Taalee. Their system has three main parts: (1) an ontology definitional component called WorldModel and assertional component called Knowledgebase, with extraction agents which manage the Knowledgebase by exploiting trusted knowledge sources. (2) Metadata extraction agents which classify and extract metadata from

content and store the result in Metabase. (3) Semantic search engine which processes semantic queries and supports limited inferencing based on the traversal of relationships in the Knowledgebase.

Guha et al. [60] augment search results by utilizing structured information from the Semantic Web. They propose a minimalist query interface called GetData which takes queries in the format of a resource and its properties, and returns a graph which contains the resource (whose properties are being queried) along with the arcs specified in the query and their respective targets/sources. They store structured data in RDF files. They also mentioned possible ways for disambiguation of search terms, such as the popularity of the terms, user profile, and search context (query history).

Ontology ranking

If multiple ontologies are used in semantic search, we have the problem of ranking the ontologies for a query, as they may represent different perception of the world. Arumugam et al. [61] propose a P2P Semantic Web approach to find relevant set of ontologies for a query. Their system matches the keywords in the query to the subjects, objects and predicates in each ontology, and discards the ontologies matching none of the keywords. Then it compares the ontologies for common parents and eliminates the ontologies without any common links. After the relevant set of ontologies is obtained from the P2P network, the personalization techniques rank the results according the user's profile and query history. The common terms in the selected ontologies may be used to explore inter-ontological relationships as well.

Alani et al. [62] present a system, AKTiveRank, for ontology ranking. Their ranking approach combines Class Match Measure (string match to class labels), Centrality Measure (the depth in the hierarchy), Density Measure (number of relationships of a concept), and Semantic Similarity Measure (shortest path measure for concepts [50]). Their evaluations indicate that Centrality Measure and Density Measure have more significant contribution to the result than the other measures.

OntoSearch [63] combines Google Web APIs with a hierarchy visualization technique and allows the user to perform keyword searches on certain types of ontology files, and to visually inspect the files to check their relevance. However, they do not provide automatic ranking for the search result.

CHAPTER 4

OVERVIEW OF PROPOSED APPROACH

Based on the comparison of folksonomy and ontology in Chapter 2, we propose an information indexing and retrieval approach that takes the advantages from both folksonomy and ontology.

First of all, we want to retain the easy-to-use nature of folksonomy. Our users should be able to continue to use tags to describe their information, and use keywords to retrieve the information they want. We do not expect them to know anything about Semantic Web, or even taxonomy. We do not, of course, expect the users to input a semantic query such as SPARQL [64]. Semantic Web technologies are completely transparent to our users. Figure 1 is an abstract overview of our approach.

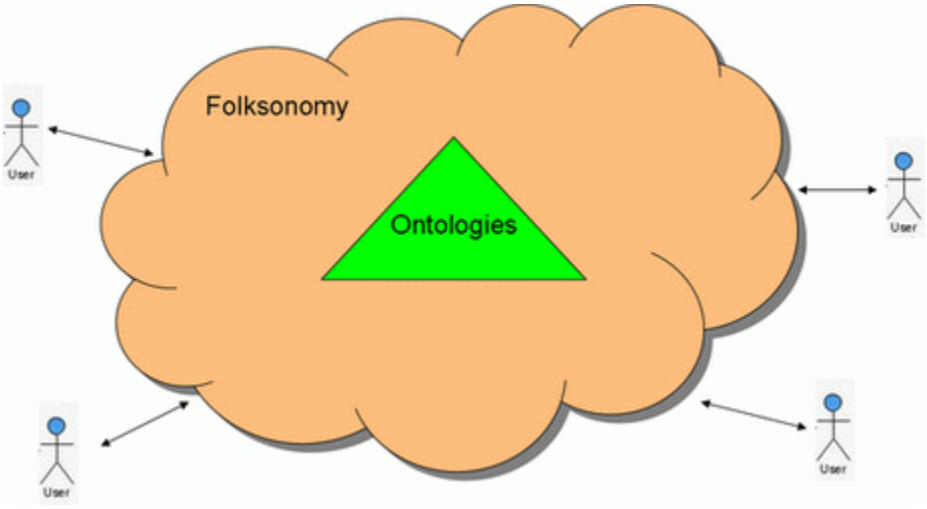

Figure 1. Abstract Overview of Combining Folksonomy and Ontology

This keyword-based approach is inevitably followed by the problem of word ambiguity. We try to overcome it, or at least alleviate this problem to a significant extent. A user should be able to tell the system by a convenient way that which sense he/she refers to for each ambiguous keyword. Then the system should show the information relevant to the selected sense on the top of the search result.

In order to distinguish the same tags with different senses, naturally, we want to index Web resources by senses, instead of the strings of tags. However, keywords are not one-to-one corresponding to senses. As humans, we tend to use keyword; but for computers, they want everything unambiguous. Figure 2 illustrates an example. The word “ticket” may have more than one meaning (termed polysemy), such as a ticket for entering a park, or a ticket for speeding. On the other hand, both “ticket” and “fine” may have the same meaning. This is called synonym.

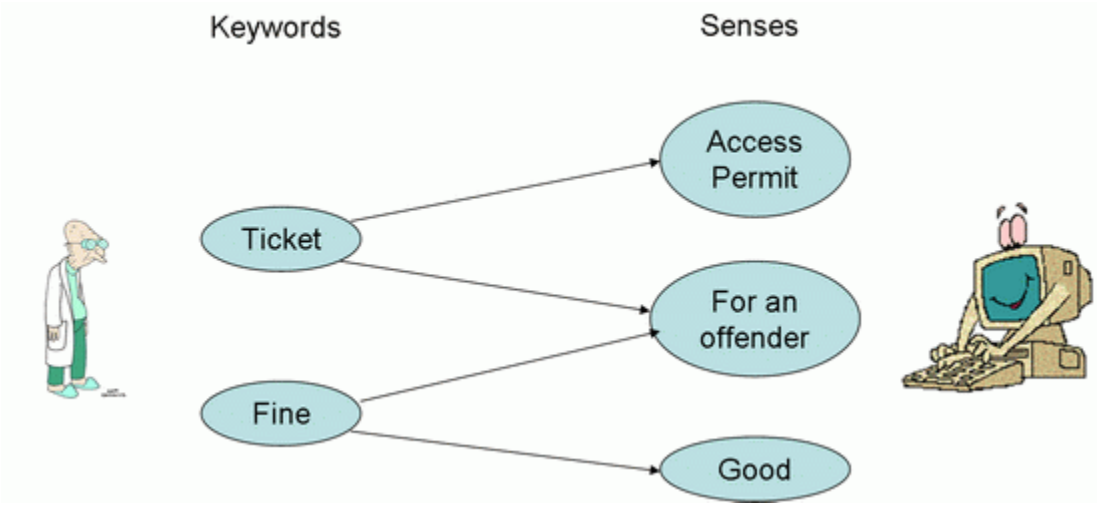

Figure 2. Relations between Keywords and Senses

Our approach to deal with the problem of ambiguity has two steps: first, we extend any keyword (both in indexing and query) to a set of keywords which are the synonyms of the original one. This is done by looking up WordNet [43]. Let us call any two keywords **equivalent** if their synonyms have overlaps. Note that equivalent keywords may have different meanings. Then in indexing, we try to cluster equivalent keywords based on their meanings by comparing their contexts. A context of a tag is the other tags co-occurring in the same Web resource. As a result, tags with the same (or similar) senses are put into the same cluster (we call them sense clusters). However, this automatic disambiguation approach does not match perfectly to our perception of the world: an actual sense may be represented by more than one sense cluster. This is because our system does not have comprehensive knowledge like a human, and even for humans, the distinctions of senses of some words are subtle. We show in figure 3 an example of disambiguating “turkey” from our datasets. Each cluster is shown by the most popular tag in it, and the edges are the relatedness (we will discuss in Chapter 6) of the clusters. The left part of this figure is about animal turkeys, the middle part is about hunting turkeys in Nebraska, USA, and the right part is about traveling in Turkey.

Figure 3. An Example of Sense Clusters

Our sense construction and disambiguation process is different from most work in Word Sense Disambiguation (WSD) such as [41, 42, 46, 47, 65], where the contents being processed are most likely documents and each keyword is in a sentence. In our case, we only have individual tags. That means there is no sentence structure or part-of-speech analysis that we can use, and the order of the tags are not necessary relevant. That the tags are produced and consumed in a social context also means significant number of terms are not in lexicons, thus we need to define the sense for those terms. The good news is that there is relatively less noise (although spam tags exist).

After we group the tags into clusters, we match each cluster to ontological concepts (if appropriate) in the ontologies in a repository. This is done automatically by comparing the context of a cluster and the related concepts of an ontological concept. Figure 4 illustrates this idea.

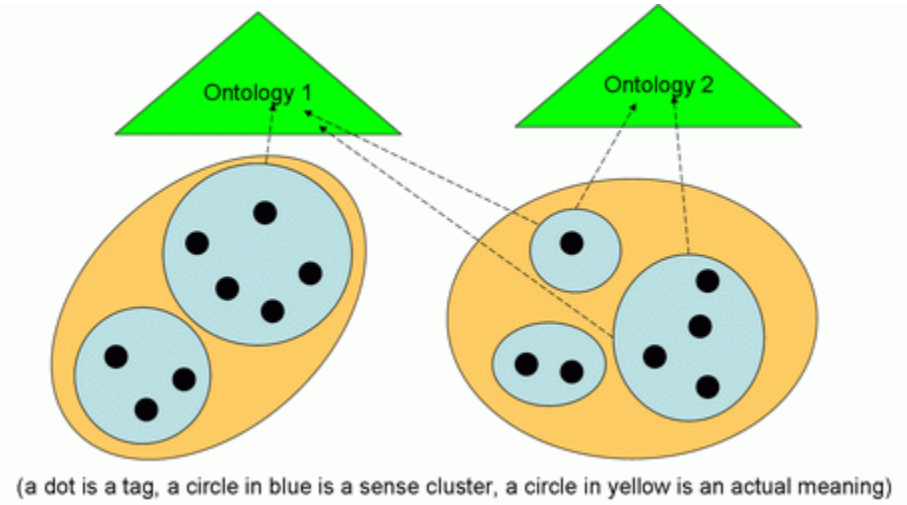

Figure 4. Sense Clusters and Ontology Matching

To enable semantic search capability, our system extends query keywords to include their synonyms, and then locates candidate sense clusters by both (a) matching the query keywords with the tags in sense clusters, and (b) performing semantic query on ontologies in a repository and find corresponding sense clusters. This may include inappropriate sense clusters, caused by polysemy. In order to let the user elect which sense(s) he/she refers to, our system shows one sample item for each candidate sense cluster. The user may select a sample item which seems the best he/she wants, and then the systems ranks other candidate sense clusters by calculating the similarities between them and the selected one.

The result ranking may involve multiple ontologies. For example, a query of “city” may be performed on an ontology in geography domain, and another ontology in politics domain. If the cities in these ontologies are (partially) overlapped, we will have the problem about ontology we should trust, or how much we should trust. We use the query history of a user as the context of a query, and automatically assign a weight to each ontology. The ontology with higher weight has greater power to vote on the ranking of result; the ontologies with too small weight will be discarded for the query.

Figure 5 is a part of the ontology matching result produced by our system. Some sense clusters are matched to one or more ontological concepts (we call those clusters **onto-clusters**), while others do not, even though they have the same tags as in the onto-clusters.

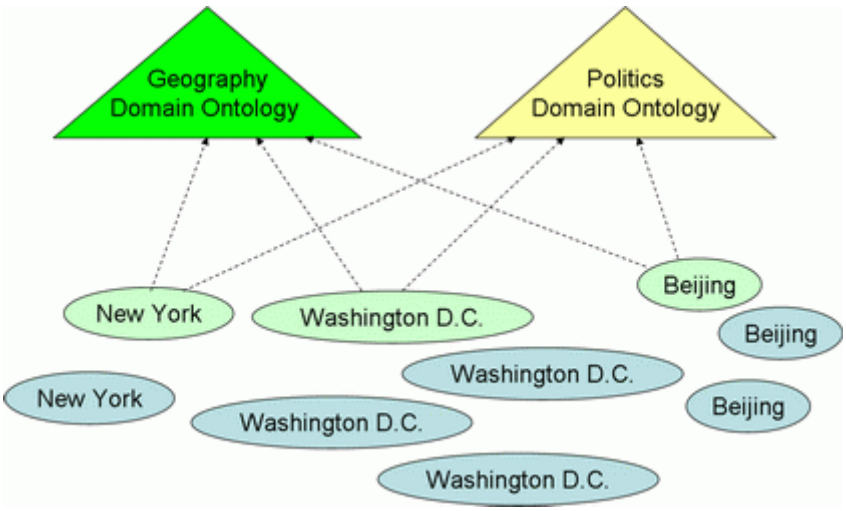

Figure 5. Multi-ontologies Matching

The overview of our system is illustrated in Figure 6. First, we clean up the data input into our system, to eliminate string matching problems, such as plural nouns and misspelling. Then we build an index based on the senses of tags. Newly created sense clusters are automatically matched to ontologies in our repository. The search engine locates candidate sense clusters and ranks the result with the help of ontologies.

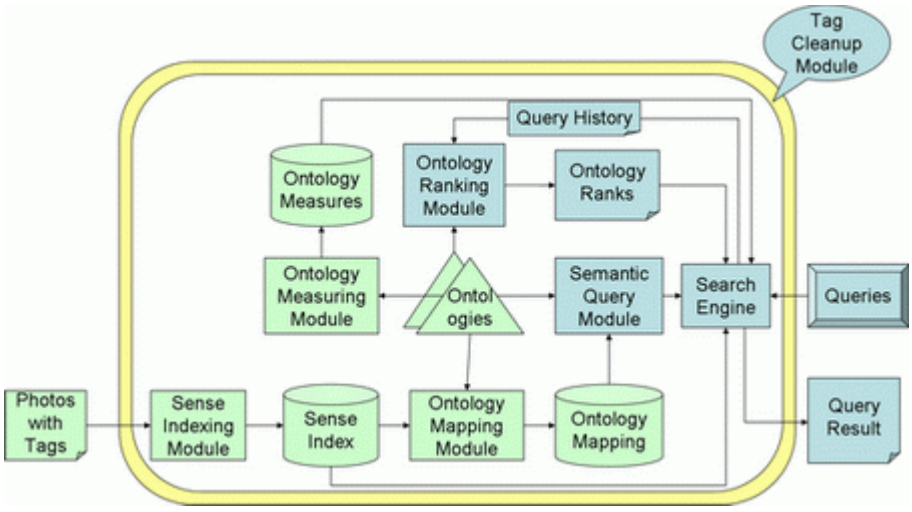

Figure 6. System Overview

CHAPTER 5

DATA CLEANUP

While tagging systems give users the freedom to choose whatever keywords they like, they inevitably introduce a problem: various versions of a keyword are used by different users, such as single and plural nouns. These different versions are represented as different strings, thus a string comparison will determine they are different. However, the users actually refer to the same meaning. This phenomenon is not synonym: it is because of different customs from different users. Our approach to this problem is to extract their patterns, and convert them into a canonical form.

As a summary, we list the usual problems we have encountered:

- Use symbols to separate words. For example, “me/projects/travelbuddy”
- Use symbols to distinguish one’s own tags. For example, “@@daily”
- Use tags for date. For example, “imported:2005-09-17”, “europe2005”
- With or without “-”. For example, “ebook” and “e-book”, “air-ticket” and “airticket”
- Two or more words are represented as one word. For example, “freephotos”, “travelagent”
- Misspelling. For example, “sculture” sould be “sculpture”
- Plural noun and verb tense. For example, “bird” and “birds”

We call the procedure of converting tags to their canonical forms tag normalization. The objectives are:

- The tags with the same meaning should be represented as exactly the same strings, except synonyms.
- Separate the tags such as “freephotos” into multiple tags, as each word has independent meaning.
- Separate the tags such as “travelagent” into one tag of a compound word, as users will most likely use both words in their queries to retrieve those photos.

Our normalization procedure includes three steps:

1. Detect any characters in the tag which is neither an English letter nor a number (we do not consider tags in other language than English). Use those characters as delimiters to split a tag into one or more words. There are two exceptions in this step. First, if the tag has only numbers, such as “20070801”, we discard the tag. Second, we keep a list of special tags such as “DVD-R”.
2. Obtain the canonical form of each word. We need a comprehensive mechanism which can convert any keyword to its canonical form. We use two online dictionaries Webster.com [66] and Dict.cn [67] for this task. Webster.com provides the root form of a word, and also gives suggested word for a misspelled word. For example, “swimming” is converted into “swim”, “dogs” into “dog”, and “sculture” into “sculpture”. However, Webster.com along is not a complete solution. For instance, IBM is not an entry in Webster.com, and it also has very limited compound words, such as “open source” is neither an entry. Dict.cn

provides more abbreviations and compound words. Therefore, the system checks both dictionaries to determine the canonical form a tag.

3. If both dictionaries have no entry for the word we give for check, we consider it is a compound word with white space omitted by the user. Then we try to split the word into two or more, and check each word as in step 2. For the user invented tags which are in neither dictionary, we just keep it untouched.

CHAPTER 6

INDEXING BY SENSES

1. Introduction

Currently most information retrieval systems index documents by keywords, store the index in hash tables, and usually provide good performance for keyword based queries. This approach works fairly well in most cases because significant part of senses from our perception can be uniquely or approximately identified by keywords. However, the fact that keywords and senses are not one-to-one corresponding is a permanent barrier to achieve very high precision and recall rates for keyword-based search engines.

In this thesis, we attempt to break this barrier by indexing Web resources (in our case, photos) by senses instead of strings of keywords. As discussed in Chapter 4, our system need to construct senses automatically and determine which sense a keyword/tag has. Our approach is partially similar to [68], where the authors partition a large ontology into domain modules by creating Dependency Graph of the concepts, and determining the modules by finding maximal spanning tree in each module.

2. Definitions

Before we describe in detail our approach of indexing Web resources by senses, we give definitions of important terms we will use in the rest of this thesis.

- **Web resource (or resource)**: anything with a URL
- **Label:** one or more keywords, e.g. air ticket. String comparison of two labels is atomic, i.e., we do not compare a part of a label.
- **Tag:** a label tagged to a resource. Two different tags may have the same label, if there are tagged to difference resources.
- Context of a tag: the tags co-occur with a tag *t* in a resource along with the co-occurrence frequencies are called **the context of *t***. For convenience, let us define the following functions:

  fTagFreq(t): the occurrence frequency of tag *t* in a resource

  fTagCoFreq(t_1, t_2): the co-occurrence frequency of t_1 and t_2 in a resource.

  fTagContext(t_i)= {t | *fTagCoFreq* (t_i, t)>0}: the context tags of a tag t_i
- Sense Cluster (or cluster): where tags with similar meanings are put together. A tag must belong to exactly one cluster.
- Context of a cluster: a set of other clusters (called context clusters) with cluster co-occurrence frequencies. The co-occurrence frequency of two clusters is the aggregation of the co-occurrence frequencies of the tags in the clusters. Let us define the following functions:

  $fClusterCoFreq(c_1,c_2) = \sum_{t_i \in c_1, t_j \in c_2} fTagCoFreq(t_i,t_j)$: the co-occurrence frequency of c_1 and c_2.

  $fClusterContext(c_i) = \{c \mid fClusterCoFreq(c_i,c) > 0\}$: the context clusters of a cluster c_i.
- Relatedness of clusters: this is to measure how important a cluster is to another cluster. We define it in a similar way as TF-IDF (term frequency–inverse document frequency) weight in

Information Retrieval. Note that this measure is asymmetric. The relatedness of cluster c_j to cluster c_i is:

$$\text{Re}latedness(c_j, c_i) = tfidf_{c_j \to c_i} = tf_{c_j \to c_i} \times idf_{c_j}$$

$$tf_{cj->ci} = \frac{fClusterCoFreq(c_i, c_j)}{\sum_{cx \in C, cx \neq ci} fClusterCoFreq(ci, cx)}$$ (C is the set of all clusters)

$idf_{cj} = \log \frac{|C|}{|\{c : fClusterCoFreq(c, c_j) > 0\}|}$, where $|C|$ is the total number of the clusters in the system and the denominator is the number of clusters where at least a tag in it co-occurs with any tag in c_j.

Let us define the function *fRelatedness(c_j, c_i)* to return the relatedness of c_j to c_i.

- Semantic Annotation: a set of URIs which associatesa cluster with ontological concepts.
- Important context of a cluster: if we sort the context clusters of a cluster *c* by their relatedness to *c*, the top k% (k is a parameter) context clusters are called the important context of *c*. The reasons why we need this definition are because (a) the tags of a resource are often about multiple independent objects, such as “boy” and “house”, where they just happen to be in the same photo, and (b) users may input whatever tags to describe a resource, thus some tags are frequently used and thus important, while others are not. This phenomenon can be represented by The Long Tail [69] pattern. By only taking the top k% context clusters, we concentrate on the important tags which are most likely the signs for determining a sense, and filter out noises. We may further divide the important context into three levels, as shown in Figure 7. Different levels will be used in different phases in building senses, which will be discussed in section 3 of this Chapter.

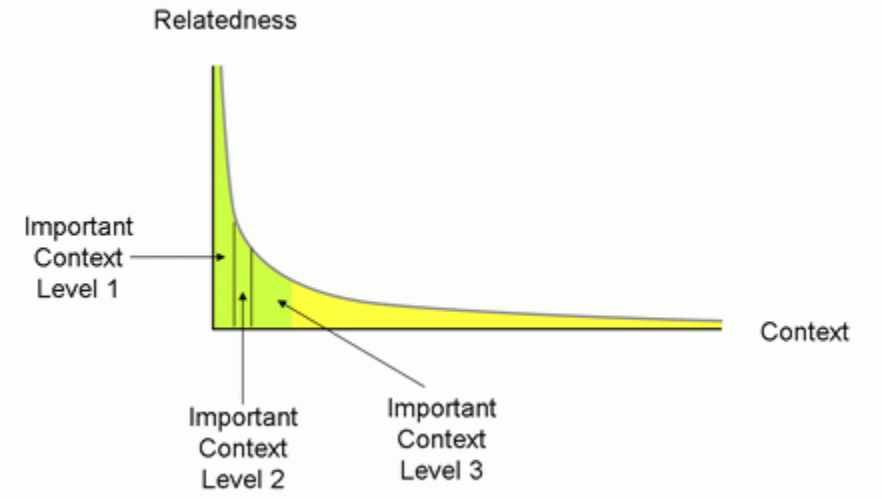

Figure 7. Important Context

2. Synonym

Synonym is the phenomenon that two or more words or expressions of the same language have the same or nearly the same meaning in some or all senses. Abbreviation can be regarded as a special type of synonym.

The problem caused by synonyms in Information Retrieval is that information publishers may use keyword kw_1 to describe a resource, while information consumers may use keyword kw_2 to search the resources, where kw_1 and kw_2 are synonyms but are represented as different strings, thus systems fail to retrieve the resources described by kw_1. This results in low recall rate in information retrieval. For example, a query of “United Kingdom” fails in returning a photo with a tag “UK” but without a tag “United Kingdom”.

The usual solution for synonyms is to expand a keyword to a set of keywords by looking up a synonym list. We adopt WordNet [43] for this purpose. WordNet is a semantic lexicon for the

English language. It groups English words into sets of synonyms called Synsets, provides short, general definitions, and records the various semantic relations between these synonym sets. For each tag to be indexed and each keyword in queries, we expand it into a set of keywords by locating a set of Synsets in WordNet, and returning all the synonyms in those Synsets.

An interesting research problem is that if we can detect synonyms automatically, without depending on a synonym list. For folksonomies, we believe it is impossible to have satisfactory result. The reasons are:

1. The user may not input all the important features of a concept in the tags. For example, “feather” is an important feature of birds, but there is no garreteer that majority of the users put “feather” in their tags to describe birds.
2. After all, that the contexts of two tags are similar enough does not imply the tags are synonyms: it only indicates they are very related, for example, “apple” and “orange”.

3. Polysemy

Polysemy is the phenomenon that a word has multiple meanings. The problem caused by polysemy in Information Retrieval is that information publishers and consumers may use the same keyword to refer to different meanings, and as a result, systems return something which is unwanted. Polysemy is an important reason of low precision rate in Information Retrieval. For example, a query of “apple” may return photos about the fruit apple, and also the Apple electronic products.

Intuitively, context is the basis to determine the meaning of a tag. But there are two fundamental questions we need to consider.

Question 1: how can we define the senses for each tag?

Question 2: how can we determine which sense a tag refers to?

For question 1, one approach is to adopt a comprehensive thesaurus, such as WordNet, and match each keyword to the senses defined in the thesaurus. Some previous work [45, 65, 70] has been done in this way. However, there are at least three problems that hamper the usability of this approach.

Problem 1: The thesaurus must be comprehensive enough to include all potentially used terms. For the new meaning invented by the users, we do not have any other choices but have to create a new entry in the thesaurus. This problem becomes more severe in the fields where terms change quite often, such as in social networks.

Problem 2: Some distinctions of senses are subtle. For example, some people consider "bookmark" in Del.icio.us and a real bookmark put in a book have the same meaning, while others do not. As another example, some people care about the difference of a turkey bird and the meat of turkey, but others do not. Therefore, we are facing the problem of how detailed we should distinguish the meanings, and how can we make sure the thesaurus are fine-grained enough.

Problem 3: The definition or explanation of a sense is usually in phrases or sentences. Comparing these phrases or sentences with the context of a tag is a problem by itself. Alternatively, we may do it in a machine learning way. We may manually assign some tags to the senses, compute the frequencies of the context tags of the assigned ones, and use that information as the context of the senses. This of course involves the overhead of training.

In this thesis, we invented a new approach which automatically builds senses clusters, and assigns tags to appropriate clusters by their contexts. The number of clusters generated depends on the relatedness of the context clusters and a given threshold.

We may state our motivation for our approach in an example. In order to search photos about turkey birds, some people use “bird” besides “turkey”, some use “animal”, some use “food”, “wild”, etc. Can we include all these tags, and then use them to build a sense? The clue to recognize these tags is that they co-occur with each others more often than with other tags (which are also the context of “turkey”)

The procedure includes the following steps:

Step 1: Put all tags with the same label (or synonyms) into one cluster (called initial cluster).

Step 2: Do the following 3 phases to build senses.

Phase 1.

1. Identify Important Context Level 1
2. Create a undirected weighted graph called Context Graph, where each node in the graph is a cluster in the Important Context Level 1, and the weight of an edge is the relatedness of the two clusters (relatedness is asymmetric, we take the larger one).
3. Apply a threshold to the edges of the Context Graph, so that the graph becomes one or more disconnected component. The threshold is a constant times the average of all the edges in the Context Graph.
4. Create a sense corresponding to each component, and use the clusters in a component as the context of that corresponding sense.

Figure 8 shows the product of Phase 1 when we build senses for the tag “turkey”. The squares are the context clusters of “turkey”.

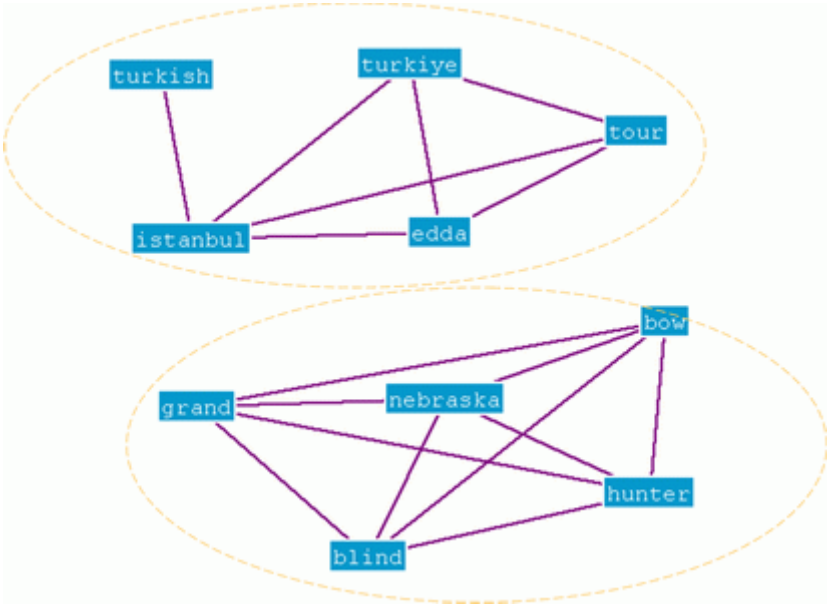

Figure 8. Product of Building Senses Phase 1

Phase 2.

Phase 1 may identify some important senses which are used quite often in the dataset. However, some important senses may be missing in the product of phase 1, because they are not used often, so their context is not in the Important Context Level 1. The purpose of phase 2 is to find those missing senses. The difference of phase 2 from phase 1 is that we do not merge the senses built in phase 1.

1. Identify Important Context Level 2
2. For each cluster in Important Context Level 2, find the most related sense built in Phase 1 (and also above a threshold).

3. If there is such a sense, merge the cluster being considered to that sense's context.

4. Otherwise, build a new sense and use the cluster as the context.

Figure 9 shows the product of Phase 2 when we build senses for the tag "turkey". The clusters circled in red are newly discovered in Phase 2.

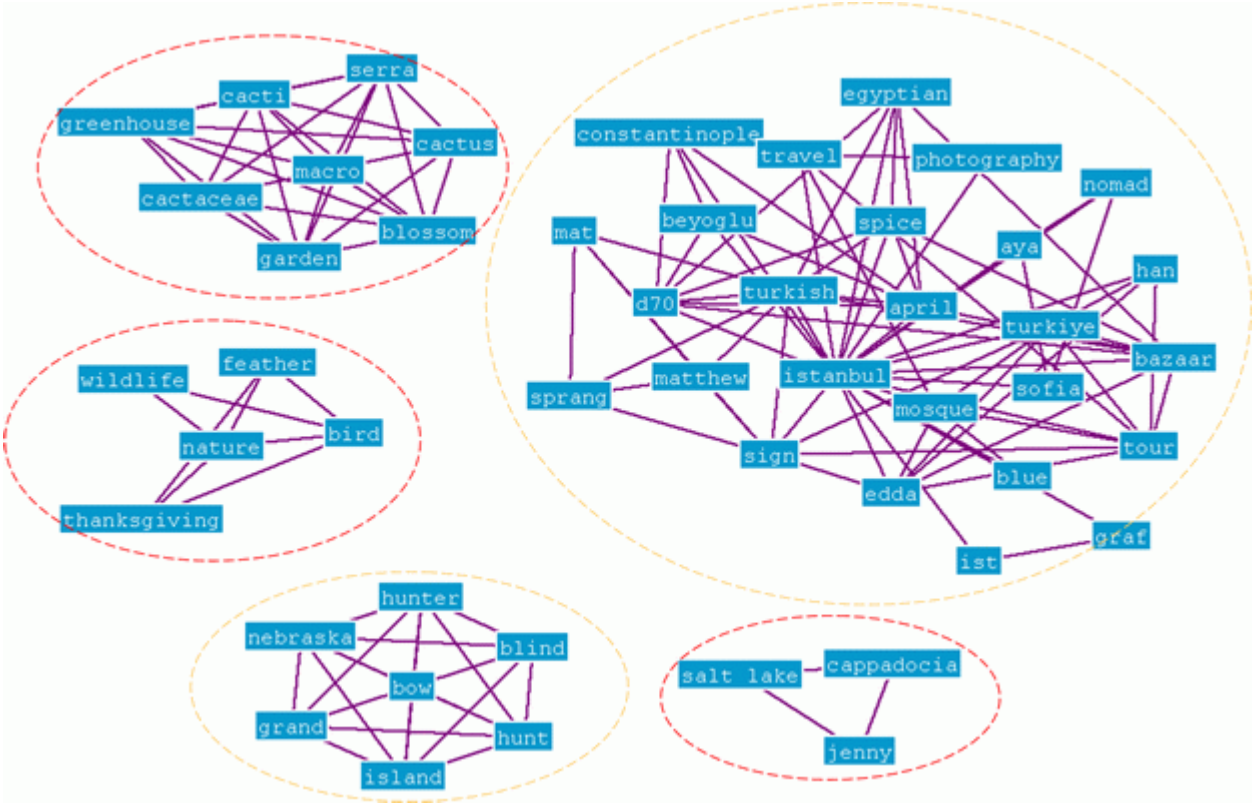

Figure 9. Product of Building Senses Phase 2

Phase 3.

In Phase 3, we identify Important Context Level 3, and we do not create any new sense, but just enrich the context of the senses built in Phase 1 and Phase 2.

Step 3: Compare each tag we are considering with the senses. Select the best matched sense and assign the tag to it.

Step 4: Do step 2 and step 3 again when the number of the tags we are considering is increased to a certain percentage. This is reasonable and also necessary, as the new information may contain new evidence that we have not had before.

Since the important contexts are the “head” of The Long Tail pattern, it guarantees that most tags are assigned to a cluster. The above procedure may generate more than one cluster for an actual sense we usually perceive. Therefore, we need to rank these clusters in the search engine. This issue is discussed in Chapter 8.

Let us consider the second question: even though we have built the senses, how can we determine which sense a tag refers to? There are some well-known approaches to calculate information similarity, such as Jaccard similarity coefficient, Dice coefficient, and Information Content. Jaccard similarity coefficient is usually used to calculate the similarity of two vectors. However, the dimensions are usually not independent in the real world. For example, both “feather” and “bird” can be part of the context of “turkey”, but “feather” and “bird” are quite related. Some researchers [49] have proposed some variants of Jaccard similarity coefficient in order to take into account the dependency of different dimensions. Another issue that we should pay attention is that a tag and a cluster are not balanced: the size of the context of a cluster is usually more than the size of the context of a tag, because a cluster consists of numerous tags.

Based on the above considerations, we compare a tag with a cluster as follows.

$$compare(tag, cluster) = \forall t \in fTagContext(tag), \forall c \in fClusterContext(cluster), \sum tf(t, tag) \mid matched(t, c)$$

$matched(t, c)$ is true if $\exists t' \in c, t.label = t'.label$

$tf(t, tag) = \dfrac{fTagCoFreq(t, tag)}{\sum_{m \in tag.resource} fTagCoFreq(m, tag)}$, where *tag.resource* means the resource *tag* belongs to.

The idea is that the more context of a tag is matched by the context of a cluster, the more likely the tag belongs to the cluster. Function *compare(tag, cluster)* aggregates the TF weights of the context tags which matches to the context of *cluster*. Function *matched(t,c)* determines if a context cluster *c* of the cluster we are considering matches to a context tag *n* of the tag we are considering. It returns true if there is at least a tag in *c* whose tag is the same as (or synonym of) the tag of *t*.

After comparing a tag with all potential clusters, we assign the tag to the cluster with highest score, if the score is greater than zero; otherwise, we mark the tag undetermined.

CHAPTER 7

UTILIZING ONTOLOGIES

1. Introduction

Ontology is a way to define concepts and their relationships explicitly. Ontology designers may formalize their knowledge in a way easier for computer to process, or more concretely, easier to write programs to utilize the knowledge. This advantage comes from the use of URIs (Unified Resource Identifier) and named relationships of concepts.

Ontologies may provide accurate information as background knowledge to folksonomies. This is especially useful for well-established knowledge, for example, ontology may state explicitly that Atlanta is a city in the United States.

However, a single ontology may not be able to provide all the knowledge for an application, thus a scalable way is to use multiple ontologies. An ontology may be domain specific, the knowledge presented by an ontology is subject to the designer's perception of the world. How to select appropriate ontologies [63] and how to rank the candidate ontologies [62] is an on-going research topic in Semantic Web community.

We use both the local name of the URI of an ontological concept and its label property defined in RDF and OWL as the keyword(s) for a concept.

2. Semantic Annotation

In order to utilize ontologies as background knowledge, we match the clusters to ontological concepts where it is appropriate. To make our approach scalable, we should be able to use multiple ontologies at the same time. Therefore, a cluster may be matched to concepts in more than one ontology. On the other hand, we assume that no two concepts in the same ontology are duplicated, thus a cluster can be matched to only one concept in an ontology. And more than one cluster may be matched to the same concept, as the clusters are generated automatically and we cannot guarantee there is no overlap of the meaning of the clusters.

A significant difference between ontology and folksonomy is that the information represented by ontologies is more explicit than by folksonomy, specifically, there is no named relationship between the tags in folksonomy. That means we cannot compare tags with ontological concepts by the names of relationships. Nevertheless, we may utilize the named relationships in ontologies to calculate the relatedness of ontological terms, and then compare with tags by the calculated the relatedness.

The matching process is similar to the comparison of tags with clusters. We compare the important context of a cluster with the context of an ontological concept. The context of an ontological concept is other ontological concepts connected to it by any relationship defined in the ontology schema. Note that a context cluster is considered matched to a context concept if any tag in the cluster has the same label/keyword as the context concept, or they are very relevant. For example, “bird” is a context of an ontological concept “turkey”, and “feather” is a context of a cluster containing “turkey”, thus “feather” should be considered matched to “bird” if

we have the evidence that they are very relevant. The evidence may come from (a) the relatedness of two clusters, (b) the relatedness of the corresponding ontological concepts, or (c) the similarity of the corresponding ontological concepts, which will be discussed in Section 4 of this Chapter.

3. Relatedness of Ontological Concepts

We define the relatedness of two ontological concepts in a very similar way as in the relatedness of two clusters. We get the co-occurrence frequency of each pair of the concepts by issuing queries to Yahoo! Search engine and using the hit count, thus we may calculate the Term Frequency (TF) weight of any pair of ontological concepts. For the calculation of Inverse Document Frequency (IDF), we divide the hit count of a term by the number of total pages indexed by Yahoo! (currently about 20 billion), and then take negative logarithm. The relatedness of any pair of concepts is the product of TF and IDF. Note that if two concepts are not connected by any relationship, we consider their relatedness is zero.

4. Similarity of Ontological Concepts

A concept may be considered as a set of features (or properties). For different domain or different purpose, the importance of each feature may vary. An ontology usually has subclass relationships, which form a taxonomy. When the designers organize the concepts into a taxonomy, he/she actually has determined which feature is more important than the others: the distinction of the most important feature is the criterion for grouping concepts on the first level of the taxonomy. Therefore, the structure of the taxonomy of an ontology is a very important indication of the similarities of concepts.

Some research has been done in studying the similarity in a taxonomy. A straightforward way is to count the number of edges between two concepts [50]. However, it relies on the notion that links in the taxonomy represent uniform distances, which is usually not the case. Resnik [53] presents a measure of semantic similarity based on the notion of information content, which gives good experimental results. His approach considers both the structure of a taxonomy, and the informativeness of each term in the taxonomy. He defines the similarity of two concepts as the information content of their most specific common ancestor. Lin [54] normalized the similarity by taking into account the information content of the concepts being considered. We adopt his approach to calculate the similarity in a taxonomy.

We calculate the probability of each concept by issuing queries to Yahoo! Search Engine, and divide the hit counts by the number of total pages indexed by Yahoo!. We define and calculate the following numbers:

f(c): the Yahoo! Search hit count of a concept *c*.

fTotal(c): the Yahoo! Search hit count of a concept *c* and its subclasses and instances if there is any.

All: the number of total pages indexed by Yahoo!

p(c): the probability of encountering concept *c* in the dataset, which is $p(c) = \frac{fTotal(c)}{All}$

-log p(c): the information content of a concept.

We define the similarity of two concepts in a taxonomy as:

$SimTax(c_1, c_2) = \frac{2 \times \log p(c)}{\log p(c_1) + \log p(c_2)}$, where *c* is the most specific common ancestor of c_1 and c_2

However, an ontology is more than a taxonomy: it may has named relationships other than *subclass* and *type*. These named relationships represent other features / properties of the concepts which are not reflected in the structure of the taxonomy. They are also important factors influencing the similarities. For example, Atlanta, Chicago, and Beijing are all instances of a class "city", but we may have the following triples:

- *Atlanta located_in USA*
- *Chicago located_in USA*
- *Beijing located_in China*

This indicates that we should increase the similarity of Atlanta and Chicago.

We adopt Jaccard similarity coefficient for this purpose. We define:

- *relation(c)*: the triples where concept *c* is the subject and the predicate is not *subclass* or *type*
- *obj(t):* object of a triple *t*
- *matched(t1, t2)* is true if *t1* and *t2* have same predicate and object

$$Sim\,\mathrm{Re}l(c1,c2) = \frac{\sum_{t1\in relation(c1),t2\in relation(c2)} \mathrm{Re}latedness(obj(t1),c1)\times \mathrm{Re}latedness(obj(t2),c2) \mid matched(t1,t2)}{\sqrt{\sum_{t1\in relation(c1)} \mathrm{Re}latedness(obj(t1),c1)^2 \times \sum_{t2\in relation(c2)} \mathrm{Re}latedness(obj(t2),c2)^2}}$$

We combine this measure with the above similarity measure of taxonomy by using a factor *b* as a weight.

$$OntoSim(c1,c2) = \frac{b\times SimTax(c1,c2) + Sim\,\mathrm{Re}l(c1,c2)}{1+b}$$

CHAPTER 8

CONTEXT-AWARE MULTI-ONTOLOGIES SEMANTIC SEARCH

1. Semantic Search

Semantic search retrieves information based on the relationships defined in ontologies, thus breaks a barrier in keyword-based search and achieve higher recall and precision rates. For example, an ontology may advise the search engine that Atlanta and Boston are cities, thus search engine may expand a query of “city” to include these two cities.

We may consider other types of relationships defined in schemas than *subclass* and *type*. For example, we may find all the classmates of *John* and also classmates of *Mary*. SPARQL [64] support even more complex semantic search queries. We may enable reasoning to infer new knowledge from the existing knowledge in ontologies as well.

However, we do not expect our users know semantic query syntaxes, even though they do, we do not expect they will have the patience to input complex semantic queries. Therefore, we only consider *subclass* and *type* relationships in the current version of our search engine.

Figure 10 is the pseudo code of our semantic search process which returns two sets of clusters. Ranking of these clusters is not included in the pseudo code. After this process, the system may find resources by Jaccard similarity coefficient based on the returned clusters.

```
SemanticSearch (keywords)
    define a set onto-clusters
  define a set candidate-clusters
  for each keyword k in keywords
    //find ontological concepts whose keyword(s) is k or are synonym of k
      ontoConcepts←findConceptsByKeyword(k)
      if ontoConcepts is not empty
      for each c in ontoConcepts
        //include subclasses and instances of c
          expOntoConcepts←subsumption_semantic_search(c)
          for each expC in expOntoConcepts
            //find clusters with semantic annotation to expC
            ontoCluster←findOntoCluster(expC)
            add ontoCluster to onto-clusters
            //get the keywords of expC
            labels←getKeywords(expC)
            //get the synonyms of the keywords of expC
            syns←getSyns(labels)
            //find clusters with a tag whose label is the same as any word in syns
            clusters←findClustersByTag(syns)
            add all cluster in clusters to candidate-clusters
      else
      //get the synonyms of the keyword in the query
      syns←getSyns(k)
      //find clusters with a tag whose label is the same as any word in syns
      clusters←findClustersByTag(syns)
      add all cluster in clusters to candidate-clusters
    return onto-clusters and candidate-clusters
```

Figure 10. Semantic Search Algorithm

This algorithm first locates ontological concepts by their keywords. Then it expands those concepts to include their subclasses and instances, and results in a set of concepts named *expOntoConcepts*. For each concept in this set, it tries to find two types of clusters: (1) the clusters with semantic annotation to the concept (we name them *onto-clusters*); (2) the clusters with a tag whose label is the same as (or a synonym of) the keyword(s) of the concept (we name

them *candidate-clusters*). Note that *onto-clusters* is a subset of *candidate-clusters*. If there is no ontological concept matched to an input keyword, we find the candidate clusters by using the synonyms of that keyword.

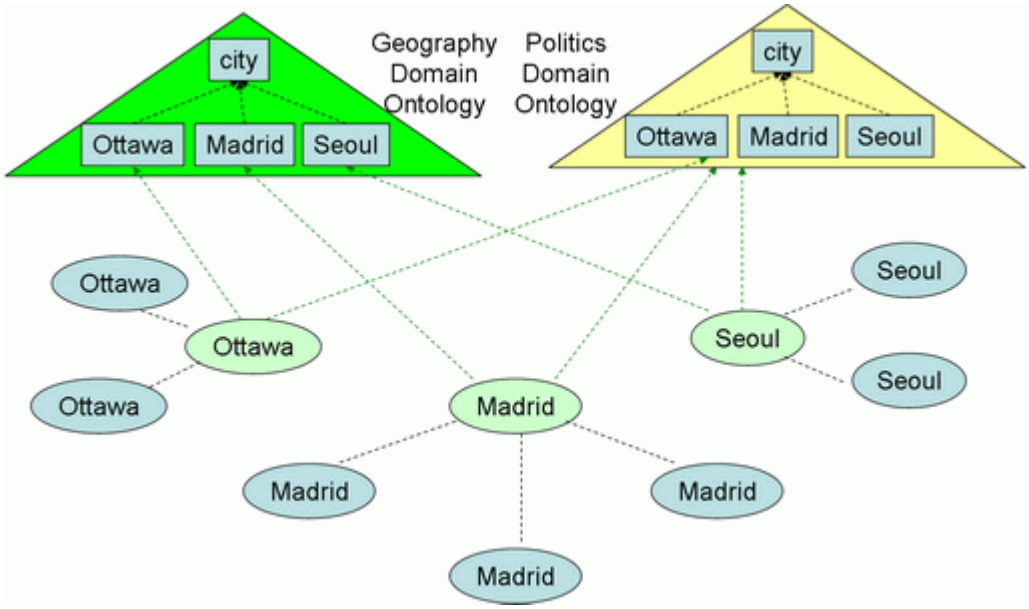

Figure 11. Example of Semantic Query

Figure 11 illustrates the process when a user issues a query “city”. Both the geography domain ontology and the politics domain ontology have a concept named city, and there are some clusters matched to their instances. We only list three cities (Beijing, New York, and Washington D.C.) in the figure for illustration purpose. The system also finds other clusters whose tags are one of the names of these cities.

2. Most-Desired Sense Ranking

After we get *onto-clusters* and *candidate-clusters* from the algorithm in the previous section, we may find the Web resources whose tags are in these clusters. However, it naturally involves

these problems: How to rank these clusters? Who is more important than the others? How to rank the result (photos)? In addition, *candidate-clusters* may include polysemy. For example, the *candidate-clusters* of a query “city” may includes clusters of “Athens”, some of which refer to the capital of Greece, while some refer to the city in Georgia, USA.

We need to rank the *candidate-clusters* so that the clusters the user concerns more have higher ranking score. Naturally we need the user to tell us which sense he/she refers to in order to rank the candidate clusters. Our approach is to show the user one item (photo) for each candidate cluster, and let the user to choose the item he wants most. We call this approach Most-Desired Sense Ranking. We convert this ranking problem to the single source shortest paths problem in a graph by doing the following:

- Build an undirected graph, whose nodes are the clusters in *candidate-clusters*, and the weight of an edge is the similarity of the corresponding clusters. We will describe how to calculate the similarity of the two clusters later.
- The cluster chosen by the user become the source in the graph.
- Instead of finding a shortest path to each node from the source, we put a constant energy in the source (for example, 1.0), and distribute the energy to other nodes. Let us define how the energy goes thought the edges. When an energy valued e goes through an edge whose weight is w, the energy’s value becomes $e \times w$. Distributing the energy through the graph is essentially equivalent to finding shortest paths, because any edge in the graph has a weight from 0 to 1, thus the energy gone through two edges one after another must be less than that gone through any one of the two edges.

Distribute the energy as described above, record the energy each cluster gets, and use that to rank the candidate clusters. Clusters receiving higher energy have higher ranking score. For those clusters receiving zero energy, we give it a very small energy, in order to not exclude them in the result.

Figure 12 shows what happen if the user searches for “city” and selects “Madrid” as the most desired cluster. The numbers besides each cluster is the energy they get.

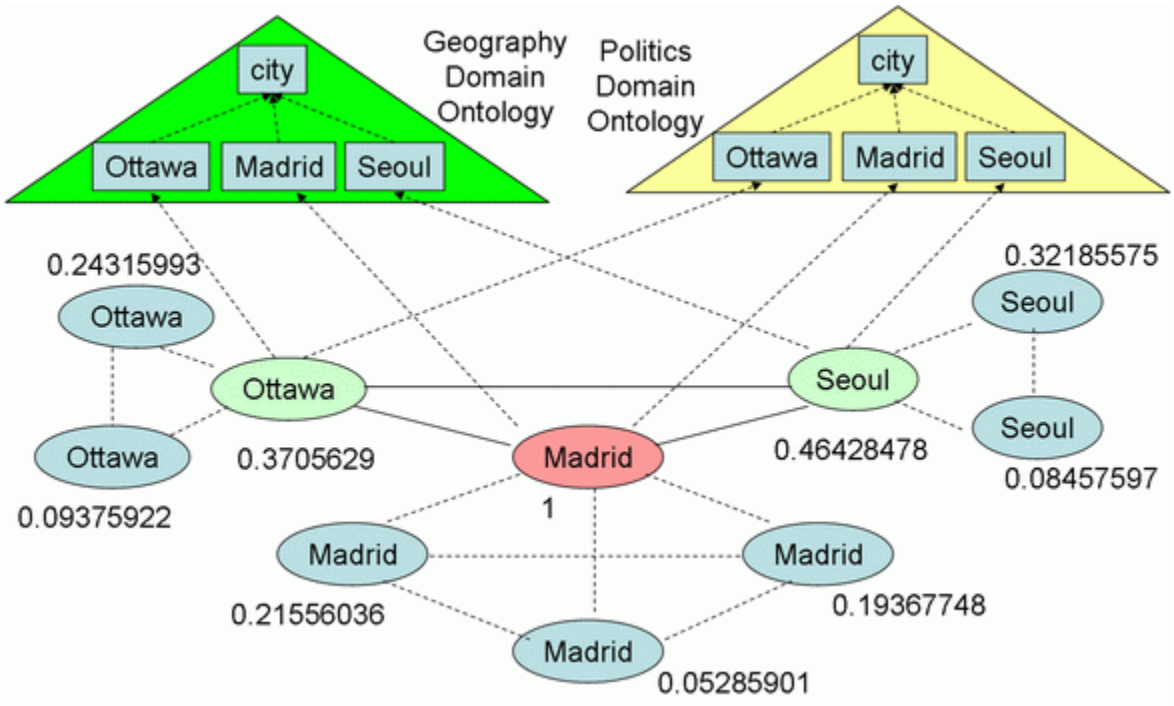

Figure 12. Example of Most-Desired Sense Ranking

The weight of the edges in the above graph is the similarity of two clusters. For calculating the similarity of clusters, we take the advantage from ontologies, as ontologies usually provide reliable background knowledge. Remember that *SemanticSearch* algorithm also returns another

set of clusters: *onto-clusters*, which is a subset of *candidate-clusters*. If the two clusters we are considering are both in *onto-clusters*, we use the similarity of the corresponding ontological concepts. We have described how to calculate the similarity of ontological concepts in Section 4 of Chapter 7.

However, a cluster may be matched to concepts in more than one ontology. For example, “city” may be a concept in an ontology in geography domain, and another ontology in politics domain. Then different ontologies may have different similarity for two concepts, thus correspondingly, two clusters. We will describe ontology ranking in the next section.

If either cluster is not in *onto-clusters*, we calculate their similarity by using their context clusters. Again, we only use the important context of each cluster in order to filter out noises. We use Dice algorithm instead of Jaccard similarity coefficient because Dice is less computationally expensive, and more importantly, often times two context clusters may be different but should be considered matched, like in matching clusters to ontological concepts in section 2 of chapter 7.

The similarity of two clusters based on their context clusters can be calculated as follows. It is a modified version of Dice similarity.

```
ContextSim(cluster1, cluster2)
    define real number variables totalRel and totalMatchedRel
    define two sets matchedContexts1 and matchedContexts2
    for each cc1 in fClusterContext(cluster1)
        for each cc2 in fClusterContext(cluster2)
            if cc1 and cc2 is matched
                add cc1 to matchedContexts1
                add cc2 to matchedContexts2
    for each cc1in fClusterContext(cluster1)
        add fRelatedness(cc1, cluster1) to totalRel
        if cc1∈ matchedContexts1
            add fRelatedness(cc1, cluster1) to totalMatchedRel
    for each cc2 in fClusterContext (cluster2)
        add fRelatedness(cc2, cluster2) to totalRel
        if cc2∈ matchedContexts2
            add fRelatedness(cc2, cluster2) to totalMatchedRel
    contextSim←0
    if （totalRel>0）
    contextSim←totalMatchedRel / totalRel
    return contextSim
```

Figure 13. Context-Based Cluster Similarity Calculation Algorithm

3. Context for Ontology Ranking

Since we may use multiple ontologies at a time, we need to rank the ontologies, and give a weight to each one. Alani et al. [62] rank ontologies for a query (a set of keywords) by combining Class Match Measure, Centrality Measure, Density Measure, and Semantic Similarity Measure. We adopt the Centrality Measure, Density Measure, as the other measures have not significant effect on the ranking result. In the following, we first describe how to rank ontologies based on one keyword, and then a query, i.e. a set of keywords, and finally include previous queries.

For a keyword *k* in a query, we find a set of ontological concepts, whose keyword(s) are the same as *k* or are synonyms of *k*, then for each concept in this set, we calculate their concept ranking score, thus come up with the ontology ranking score for an ontology.

$ontoRank(o,k) = \frac{\sum_{c \in matched(k,o)} conceptRank(c)}{|matched(k,o)|}$, where *matched(k, o)* is the set of concepts in ontology *o* whose keywords are the same as *k* or are synonyms of *k*.

We calculate the concept ranking score of a concept *c* as:

$conceptRank(c) = \frac{w_{cem} \times cem(c) + w_{dem} \times dem(c)}{w_{cem} + w_{dem}}$, where cem(c) and dem(c) are the Centrality Measure and Density Measure of concept *c* in the ontology, respectively.

We define the Centrality Measure of a concept in the same way as in [62].

$$cem(c) = 1 - \left| \frac{D(c) - \frac{H(c)}{2}}{\frac{H(c)}{2}} \right|,$$

where *D(c)* is the hierarchical level of concept *c*, and *H(c)* is the length of the longest path from the root of the branch that contains concept *c* to its bottom node. This measure has a range from 0 to 1.

And we define the Density Measure of a concept as follows. Note that *directRelations(c)* includes relations pointing to and from *c*. *w1* to *w5* are weight factors.

$$dem(c) = w1 \times |directRelations(c)| + w2 \times |superclasses(c)| + w3 \times |subclasses(c)| + w4 \times |instances(c)| + w5 \times |siblings(c)|$$

We want to normalize this measure to a range from 0 to 1 as well. We get the minimum and maximum Density Measure in an ontology, and compare the Density Measure of *c* to them.

$demNormalized(c) = \frac{dem(c) - \min_{1 \le i \le n} dem(c_i)}{\max_{1 \le i \le n} dem(c_i) - \min_{1 \le i \le n} dem(c_i)}$, where n is the number of concepts in the ontology.

For a query with a set of keywords K, the ontology ranking score is:

$$ontoRank(o, K) = \frac{\sum_{k \in K} ontoRank(o, k)}{|K|}$$

Finally, we not only rank ontologies based on one query, we also take into account the other previous queries, i.e. the query history of a user. We define a query history of a user *u* as:

QHistory(u)={<Query, Weight>}, where *Query* is a set of keywords.

A more recent query has higher weight. In the current implementations, we record up to 4 most recent queries (not including the current query), and assign weights 0.5, 0.4, 0.2, and 0.1 to them. The final ontology ranking score is a weighted average of these queries.

As an example, when the query history of a user is “cave, river, atmosphere, mountain”, the similarity of “New York” and “Washington D.C.” is 0. 5247488, while the similarity of “Beijing” and “Washington D.C.” is 0. 16578797. Contrastingly, when the query history of a user is “governor, organization, election, president”, the similarity of “New York” and “Washington D.C.” is 0.08279617, while the similarity of “Beijing” and “Washington D.C.” is 0.46662706.

4. Context for Keywords

Besides the Most-Desired Sense Ranking approach we described above, we also provide another way to promote some clusters in the result. The users may put a set of keywords (we called them **query keyword’s context**) between a pair of parentheses following a keyword in a query, for example, “city(Europe)”. The search engine promotes the candidate clusters who are

related to the keyword(s) between the parentheses. This is useful not only for the final result, but also for showing the sample items for disambiguation. For example, it is hard for users to select a most-desired city from numerous cities in the world. For this purpose, we utilize the relatedness of clusters, and also the relatedness of corresponding ontological concepts, if there is any.

The following algorithm describes how to decide the extent of promotion of the clusters of a keyword. It locates the clusters and ontological concepts of the query keyword's context by string comparison, and finds the maximum relatedness of those clusters to the candidate clusters. *p* is a factor to decide the overall extent of this type of promotion.

```
Promote (candidateClusters, queryKeywordContext)
    for each candidateCluster in candidateClusters
        define real numbers maxRel1, maxRel2, totalRel and avgRel
        //find corresponding ontological concepts
        candidateOntoConcepts←findConcepts(candidateCluster)
        for each qkc in queryKeywordContext
            //get the synonyms of qkc
            syns←getSyns(qkc)
            //find corresponding ontological concepts
            ontoConcepts←findConceptsByKeyword(syns)
            maxRel1←findMaxOntoRel(candidateOntoConcepts, ontoConcepts)
            //find clusters with a tag whose label is the same as any word in syns
            clusters←findClustersByTag(syns)
            //find the max relatedness of candidateCluster and cluster in clusters
            maxRel2←findMaxRel(candidateCluster, clusters)
            if maxRel1 > maxRel2
                add maxRel1 to totalRel
            else
                add maxRel2 to totalRel
        avgRel←totalRel / number of keywords in queryKeywordContext
        increase the energy of candidateCluster by avgRel × p
```

Figure 14. Query Keyword Context Promotion Algorithm

CHAPTER 9

DATASETS, USE CASES AND EVALUATION

Evaluation measures

To evaluation the effectiveness of our approach, we compare our system with Google Desktop (in order to use the same dataset) in the following measures:

1. How much time a user has to spend in order to find the required photos. The requirement includes the content and the quantity of the photos.
2. How many clicks of the mouse a user has to do in order to find the required photos.
3. How many different queries a user has to issue in order to find the required photos. The user may change the query at any time if he feels necessary.

Note that our subjects browsed the photos in their normal speed, and make a count for each qualified photo they found.

Datasets

We collected two sets of photos from Flickr for evaluating our system's effectiveness in disambiguation and semantic search, respectively. The first set has 500 photos with a tag "apple", and another 500 photos with a tag "turkey". The second set has about 300 photos for each of the following tag: Beijing, Madrid, Ottawa, Rome, Seoul, Tokyo, Baltimore, New York, Pittsburgh, Washington D.C., Amsterdam, Florence, Venice, Athens Greece, Athens Georgia.

There are three domain ontologies in our repository:

- We built an ontology in travel domain, which is partially come from the taxonomy of RealTravel.com [71].
- We modified an AKTiveSA [72] project ontology into a geography domain ontology.
- We built an ontology in politics domain, which is partially come from SWETO [73].

They are partially overlapped, such as on city names.

Use cases

The first set of tasks is for evaluation of disambiguation. We illustrate the time they spent for finding these photos, the number of clicks they did in the tasks, and the number of different queries they issued.

Task1: find 50 photos about Apple electronic products

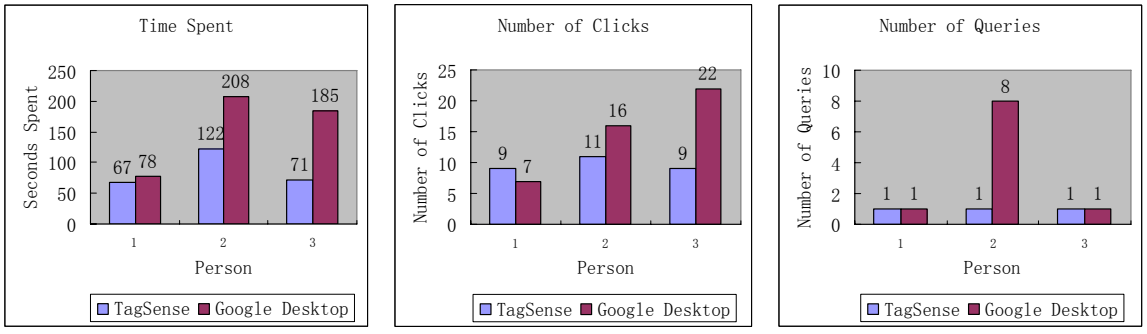

Figure 15. Evaluation of Finding Photos about Apple Electronic Products

Task2: find 30 photos about the fruit apple

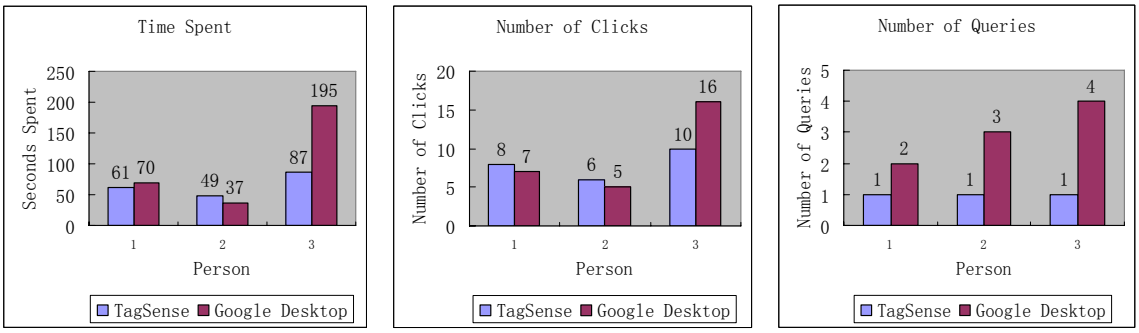

Figure 16. Evaluation of Finding Photos about Apple Fruit

Task3: find 50 photos about the country Turkey in Europe

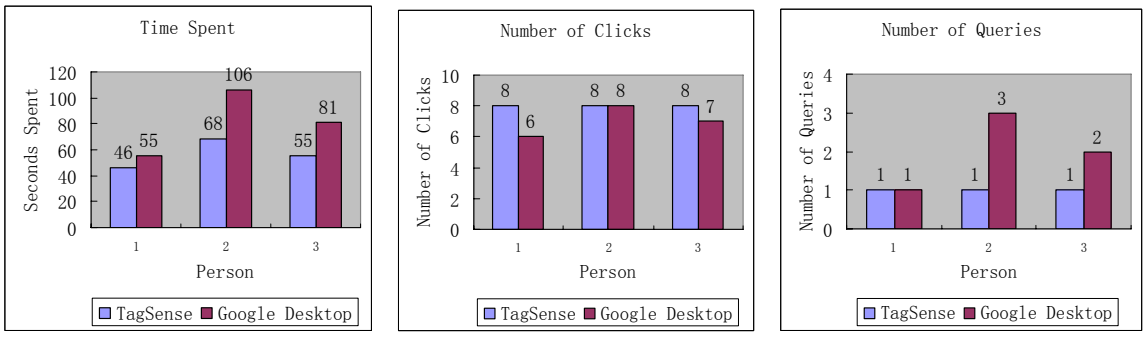

Figure 17. Evaluation of Finding Photos about Turkey in Europe

Task4: find 10 photos about turkey birds

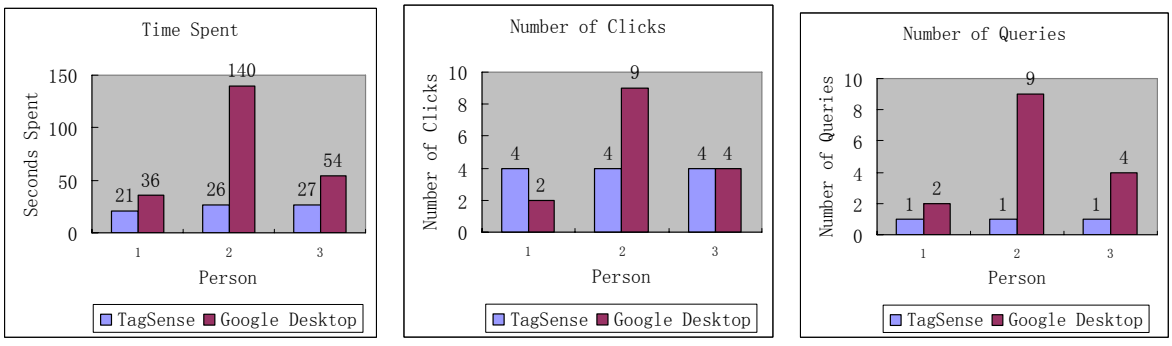

Figure 18. Evaluation of Finding Photos about Turkey Birds

We found in the experiments on Google Desktop that some uses prefer to browse the search results page by page, while others prefer to try different keywords in order to narrow the search result. Changing queries involves time overhead in thinking and typing. On the other hand, the Most-Desired Ranking approach also involves time overhead in selecting the most wanted photo among the sample photos.

The second use case is to evaluate effectiveness of our semantic search functions. We give the following task to our subjects: find up to 5 photos for each of 5 cities in Europe. We again show our results in the following graphs.

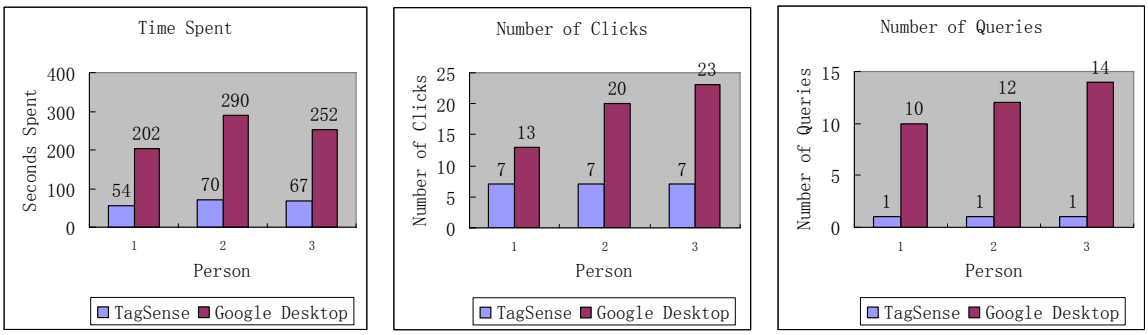

Figure 19. Evaluation of Semantic Search

This experiment is a good indication of the advantage of semantic search in the scenarios involving background knowledge. Utilizing ontologies saves much time for the users in finishing their tasks.

CHAPTER 10

CONCLUSIONS AND FURTURE WORK

Folksonomy is an easy-to-use approach to describe and retrieval Web resources. However, the ambiguity of tags and lack of explicit relationships between tags are the barriers to achieve high precision and recall rates in information retrieval in tagging systems. In this thesis, we proposed an approach to combine folksonomies and ontologies, specifically, index Web resources by senses, match them to ontological concepts, and provide semantic search capability. We also proposed Most-Desired Sense Ranking approach to rank the candidate senses, and we described how to utilize and rank multiple ontologies.

We evaluated our system by comparing with Google Desktop for the same datasets. The result indicates that our users spent significantly less time and effort in finding the wanted information on our system.

In the immediate future, we want to utilize the interest similarity of user to help tag disambiguation. For example, two users having similar interest are likely (but not necessary) use the same meaning of the same tag. We also want to connecting social networks by senses. For instance, a user seeing a bookmark of hotels in Turkey in Del.icio.us may “jump” to Flickr to see photos about the country Turkey. Last, we may integrate Geographic Information System (GIS) to our system, so that a user may, for example, search photos of rivers located with 50 miles of a given address.

REFERENCES

1. O'Reilly, T., *What Is Web 2.0? Design Patterns and Business Models for the Next Generation of Software*.
2. *Del.icio.us*. [cited; Available from: http://del.icio.us.
3. *Flickr*. [cited; Available from: http://www.flickr.com.
4. *YouTube*. [cited; Available from: http://www.youtube.com/.
5. *CiteULike*. [cited; Available from: http://www.citeulike.org/.
6. *Definition of Folksonomy from Wikipedia.org*. [cited; Available from: http://en.wikipedia.org/wiki/Folksonomy.
7. Wal, T.V. *Folksonomy Coinage and Definition*. 2004 [cited; Available from: http://vanderwal.net/folksonomy.html.
8. Gruber, T., *Ontology of Folksonomy: A Mash-up of Apples and Oranges*. International Journal on Semantic Web and Information Systems, 2007. **3**(1).
9. *Google.com*. [cited; Available from: http://www.google.com.
10. *Yahoo!* [cited; Available from: http://www.yahoo.com.
11. Gruber, T.R., *A Translation Approach to Portable Ontology Specifications.* Knowledge Acquisition, 1993. **5**(2): p. 199-220.
12. Berners-Lee, T., J. Hendler, and O. Lassila, *The Semantic Web*, in *Scientific American*. 2001.
13. Gruber, T. *What is an Ontology*. [cited; Available from: http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.
14. Group, N.W. *Uniform Resource Identifier (URI): Generic Syntax*. 2005 [cited.
15. *Panoramio*. [cited; Available from: http://www.panoramio.com/.
16. *Flickr Tour: Organize*. [cited; Available from: http://www.flickr.com/tour/organize/.
17. *Flickr: SmartSetr*. [cited; Available from: http://www.flickr.com/groups/smartsetr.
18. Marlow, C., et al. *Position Paper, Tagging, Taxonomy, Flickr, Article, ToRead*. in *Collaborative Web Tagging Workshop at WWW2006*. 2006. Edinburgh, Scotland.
19. Halpin, H., V. Robu, and H. Shepherd. *The Complex Dynamics of Collaborative Tagging*. in *WWW '07: Proceedings of the 16th international conference on World Wide Web*. 2007: ACM.
20. Wal, T.V. *Folksonomy Definition and Wikipedia*. 2005 [cited; Available from: http://www.vanderwal.net/random/entrysel.php?blog=1750.
21. Mika, P., *Ontologies are us: A unified model of social networks and semantics.* Journal of Web Semantics, 2007. **5**(1): p. 5-15.
22. Wal, T.V. *Explaining and Showing Broad and Narrow Folksonomies*. 2005 [cited; Available from: http://www.vanderwal.net/random/category.php?cat=153.
23. McGuinness, D.L. and F.v. Harmelen. *OWL Web Ontology Language*. 2004 [cited; Available from: http://www.w3.org/TR/owl-features/.
24. *Resource Description Framework (RDF)*. [cited; Available from: http://www.w3.org/RDF/.

25. *RDF Vocabulary Description Language 1.0: RDF Schema*. 2004 [cited; Available from: http://www.w3.org/TR/rdf-schema/.
26. *Ontology Inference Layer (OIL)*. [cited; Available from: http://www.ontoknowledge.org/oil/.
27. *The DARPA Agent Markup Language*. [cited; Available from: http://www.daml.org/.
28. *Standard Upper Ontology Working Group*. [cited; Available from: http://suo.ieee.org/.
29. *Simple Knowledge Organisation Systems*. [cited; Available from: http://www.w3.org/2004/02/skos/.
30. *OpenCyc*. [cited; Available from: http://www.opencyc.org/.
31. Shirky, C. *Ontology is Overrated: Categories, Links, and Tags*. 2005 [cited; Available from: http://shirky.com/writings/ontology_overrated.html.
32. Gruber, T., *Collective Knowledge Systems: Where the Social Web meets the Semantic Web.* Journal of Web Semantics, 2007.
33. Gruber, T. *TagCommons*. [cited; Available from: http://tagcommons.org/wiki/.
34. Aberer, K., et al., *Emergent Semantics Principles and Issues*.
35. Schmitz, P. *Inducing ontology from Flickr tags*. in *The Collaborative Web Tagging Workshop*. 2006.
36. Wu, X., L. Zhang, and Y. Yu. *Exploring Social Annotations for the Semantic Web*. in *The 15th international conference on World Wide Web*. 2006. New York, NY, USA.
37. Jung, J.J. and J.e.o. Euzenat, *Towards semantic social networks*, in *The 4th European Semantic Web Conference*. 2007.
38. *The Friend of a Friend (FOAF) project*. [cited; Available from: http://www.foaf-project.org/.
39. *DBLP bibliography*. [cited; Available from: http://dblp.unitrier.de/.
40. Ide, N. and J. Véronis, *Word sense disambiguation: The state of the art.* Computational Linguistics, 1998. **1**(24): p. 1-40.
41. Wilks, Y. and M. Stevenson. *Sense Tagging: Semantic Tagging with a Lexicon*. in *the SIGLEX Workshop Tagging Text with Lexical Semantics: What, why and how?* 1997. Washington, D.C.
42. Diab, M. and P. Resnik. *An Unsupervised method for Word Sense Tagging using Parallel Corpara*. in *the 40th Annual Meeting of the Association for Computational Linguistics*. 2002. Philadelphia, Pennsylvania.
43. Fellbaum, C., *WordNet: An Electronic Lexical Database*. 1998: The MIT Press.
44. Giunchiglia, F., P. Shvaiko, and M. Yatskevich. *S-Match: an algorithm and an implementation of semantic matching*. in *Semantic Interoperability and Integration*. 2005.
45. Bouquet, P., et al. *Bootstrapping semantics on the web: meaning elicitation from schemas*. in *the 15th International Conference on World Wide Web* 2006. Edinburgh, Scotland.
46. Molina, A., et al. *Word Sense Disambiguation using Statistical Models and WordNet*. in *3rd International Conference on Language Resources and Evaluation*. 2002. Las Palmas de Gran Canaria, Spain.
47. Banerjee, S. and B.P. Mullick, *Word Sense Disambiguation and WordNet Technology.* Literary and Linguistic Computing, 2007. **22**(1): p. 1-15.
48. Dong, X., A. Halevy, and J. Madhavan. *Reference reconciliation in complex information spaces*. in *the 2005 ACM SIGMOD international conference on Management of data*. 2005. Baltimore, Maryland.
49. Ganesan, P., H. Garcia-Molina, and J. Widom, *Exploiting hierarchical domain structure to compute similarity.* ACM Transactions on Information Systems (TOIS), 2003. **21**(1): p. 64-93.
50. Rada, R., et al., *Development and application of a metric on semantic nets.* IEEE Transactions on Systems Management and Cybernetics, 1989. **1**(19): p. 17-30.

51. Slimani, T., B.B. Yaghlane, and K. Mellouli, *A New Similarity Measure based on Edge Counting*. International Journal of Applied Science, Engineering and Technology, 2006. **17**.
52. Ross, S., *A First Course in Probability*. 1976.
53. Resnik, P., *Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language.* Journal of Artificial Intelligence Research, 1999. **11**: p. 95-130.
54. Lin, D., *An Information-Theoretic Definition of Similarity*, in *International Conference on Machine Learning (ICML)*. 1998: Madison, Wisconsin, USA.
55. Li, Y., Z.A. Bandar, and D. Mclean, *An approach for measuring semantic similarity between words using multiple information sources.* Knowledge and Data Engineering, IEEE Transactions 2003. **15**(4): p. 871 - 882.
56. Cilibrasi, R.L. and P.M.B. Vit´anyi, *The Google Similarity Distance.*
57. Bollegala, D., Y. Matsuo, and M. Ishizuka. *Measuring Semantic Similarity between Words Using Web Search Engines*. in *the 16th international conference on World Wide Web*. 2007. Banff, Alberta, Canada
58. Deerwester, S., et al., *Indexing by Latent Semantic Analysis.* Journal of the Society for Information Science, 1990. **6**(41): p. 391-407.
59. Sheth, A., et al., *Managing Semantic Content for the Web.* IEEE Internet Computing, 2002. **6**(4): p. 80-87.
60. Guha, R., R. McCool, and E. Miller. *Semantic search*. in *the 12th international conference on World Wide Web*. 2003.
61. Arumugam, M., A. Sheth, and I.B. Arpinar, *Towards Peer-to-Peer Semantic Web: A Distributed Environment for Sharing Semantic Knowledge on the Web*, in *International Workshop on Real World RDF and Semantic Web Applications*. 2002: Hawaii, USA.
62. Alani, H. and C. Brewster. *Ontology ranking based on the analysis of concept structures*. in *the 3rd international conference on Knowledge capture*. 2005.
63. Zhang, Y., W. Vasconcelos, and D. Sleeman. *OntoSearch: An Ontology Search Engine*. in *The Twenty-fourth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*. 2004. Cambridge, UK.
64. Prud'hommeaux, E. and A. Seaborne. *SPARQL Query Language for RDF*. 2007 [cited; Available from: http://www.w3.org/TR/rdf-sparql-query/.
65. Cañas, A.J., et al., *Using WordNet for Word Sense Disambiguation to Support Concept Map Construction*. Lecture Notes in Computer Science. Vol. 2857/2003. 2003: Springer Berlin / Heidelberg.
66. *Merriam-Webster Online*. [cited; Available from: http://www.webster.com/.
67. *Dict.cn - English-Chinese Chinese-English Online Dictionary*.
68. Stuckenschmidt, H. and M.C.A. Klein. *Structure-Based Partitioning of Large Concept Hierarchies*. in *International Semantic Web Conference*. 2004.
69. Anderson, C., *The Long Tail*, in *Wired*. 2004.
70. Fragos, K., Y. Maistros, and C. Skourlas. *Word Sense Disambiguation using WordNet Relations*. in *the 1st Balkan Conference in Informatics*. 2003. Thessaloniki, Greece.
71. *RealTravel.com*. [cited; Available from: RealTravel.com.
72. *AKTiveSA*. [cited; Available from: http://sa.aktivespace.org/.

73. Aleman-Meza, B., et al. *SWETO: Large-Scale Semantic Web Test-bed*. in *16th Int'l Conf. Software Eng. & Knowledge Eng., Workshop on Ontology in Action, Knowledge Systems Inst.* 2004.