TU Kaiserslautern
Dependable Systems Lab

Prof. Dr.-Ing. Dr. h.c. Andreas Reuter

TU Darmstadt
Ubiquitous Knowledge Processing Lab

Dr. Iryna Gurevych
Christof Mueller
Torsten Zesch

Diploma Thesis

Using Semantic Knowledge to Improve Information Search in Web 2.0

Lizhen Qu

Matrikelnummer 350 665

28.December.2007

Eidesstattliche Erklaerung

Ich versichere hiermit, dass ich die vorliegende Diplomarbeit mit dem Thema "Using Semantic Knowledge to Improve Information Search in Web 2.0" selbststaendig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnom men wurden, habe ich durch die Angabe der Quelle, auch der benutzten Sekundrliteratur, als Entlehnung kenntlich gemacht.

December 28, 2007
(Ort, Datum)

(Unterschrift)

Contents

Abstract

Ontology learning is regarded as an important step towards semantic information search in Web 2.0. This thesis presents a combined approach of NLP and machine learning to learn a domain dependent ontology from weblog and Wiktionary in terms of distributional similarity, lexico-syntactic patterns and folksonomy. Higher F-measure is achieved by our lexico-syntactic pattern based algorithm compared to the state of art text based ontology learning method. To overcome the shortage of folksonomy, a novel approach is proposed to derive a tag ontology from weblog corpus and extend it with our automatic tagging system. Efficient processing of large scale NLP application is also discussed and solutions are provided to overcome the performance bottleneck.

Chapter 1

Introduction

"Web 2.0" is one of most popular catch-up all buzzword recently that people use to describe a wide range of online applications and technologies like Wikipedia[1], weblog, social bookmarking and BitTorrent. Tim O'Reilly outlines common characteristics of Web 2.0 in (OReilly, 2005) by comparing typical examples of Web 1.0 and Web 2.0. From his point of view, Web 2.0 is a business embracing the web as a platform of applications and services, on which a software is just a single device. It has a "architecture of anticipation" that encourages users to contribute content and add value to the application in order to harness the collective intelligence. The success of Google and Overture demonstrates the collective power of a large amount of small sites by leveraging customer -self service and algorithmic data management.

The surveys of Pew Internet & American Life Project[2] show statistically an intensive usage of Web 2.0 applications. More than half of all online American youths use online social networking sites to find new friends or reinforce existing friendships. More than 36% of online American adults have looked up information from Wikipedia, whose audience grew especially dramatically in the past year from 4.0% to 20.81%. 26% Internet users have shared photos, stories and videos etc. online. 8% Internet surfers have created their own weblogs or online journals.

As more and more users are active in generating new data and sharing data with each other, it becomes a big challenge to locate and retrieval user-interested information from the web communities. Traditional Information Retrieval (IR) systems, which are based on *vector space model* (Salton et al., 1975), regard each document as a bag of words and find relevant documents by comparing the cosine similarity between a query and documents in question. Such a literal term matching strategy has severe drawbacks in real-word application. The ambivalence and synonymy of words as well as personal style and individual differences in word usage lead to unsatisfied results of document retrieval. People fail to get the most related information just because they use a synonym or another expression in query other than those in relevant documents. *Latent Semantic Analysis* (LSA) is an approach to overcome the problems by projecting queries

[1] www.wikipedia.com
[2] http://www.pewinternet.org

and documents into a low-dimensional *latent semantic space*, in which the similarities between documents or between documents and queries are more reliably estimated by mapping co-occurred terms to the same dimensions. *Probabilistic Latent Semantic Indexing* (Hofmann, 1999) (PLSA) provides a more solid statistical foundation as opposed to LSA. It allows to deal with polymous words and differentiation between different meanings and word usages. The performance of an IR system is improved in terms of better modelling of documents and queries. Another approach is closed related to Semantic Web since its emergence in late 1990's. Researchers in this area try to improve the performance of an IR system by means of an existing knowlege base or heterogeneous information sources. It allows to explore not only the co-occurrence information of terms as LSA and PLSA, other semantic relatedness can be also taken into account. KIM (Popov et al.) is a plattform capable of automatic document annotation and document retrieval based on its internal ontology. The ontology can be further enriched by named entities recognized in documents. (Castells et al., 2007) proposed a novel ontology-based information retrieval model based on an adaptation of the classic vector space model. A new scalable disambiguation algorithm proposed in (Khan et al., 2004) improves the performance of a keywords-based search system by automatically pruning irrelevant concepts and associating only the relevant ones with documents. Their works show that IR performance can be strongly improved in case of the existence of a comprehensive domain specific ontology. However, the main drawback is a lack of an domain specific ontology with high coverage, because manual ontology building demands huge amount of human efforts. So the focus of this thesis is to explore effective methods to automatically learn a domain-dependent ontology. It is especially important to exponentially growing Web 2.0 based digital assets. In this thesis, the state of art ontology learning methods are improved to learn concepts and relations from weblog posts and Wiktionary[3] glosses and higher performance is achieved by our modified approaches. Because weblog corpus is large in size, new methods are designed to be applicable to a standard workstation PC.

New Web 2.0 techniques find their own way towards better information management. Since the population of Flickr[4], del.icio.us[5] around 2004, folksonomy (also known as social classification, collaborative tagging) has been quickly gaining around as a new paradigm for web information filtering, organization and retrieval. It is a practice and method to allow web content creators and users to annotate and categorize content by means of freely chosen keywords called *tags*. As reported by Pew Internet & American Life Project, 28% of internet users have tagged or categorized web content such as photos, news stories or blog posts. Other than ontology, which is considered as hierarchical organized metadata created by information specialists, folksonomy is "metadata for the Masses" that has no hierarchy and no vocabulary control . Shirk (Shirky, 2005) points out that a large collection of individual classification schemes show a lot of strengths over traditional ontological classification in large-scale informal information sources such as World Wide Web, where most of the users are uncoordinated and amateurs. The social classification tolerates the individual categorization differences and encourages users to annotate content using their own vo-

[3] www.wiktionary.org
[4] http://www.flickr.com
[5] http://del.icio.us

cabularies. Compared to the full content of a web digital asset, the information loss by tagging is compensated as the size of folksonomy growing large because an online object can be described by various keywords from different points of view. It is especially beneficial for serendipity of browsing web content according to (Mathes, 2004) because the interlinked related tag sets are helpful to discovery unexpected web content, since tags pave different ways towards the desired content as oppose to only quite limited paths or possibilities of finding things in a static hierarchical categorized system. In a typical Web 1.0 application, people may find "Google" only through *organization* → *company* → *Google*, while in a tagging system, "Google" can be found by *search engine* → *Google*, *web* 2.0 → *Google* and so on.

Although folksonomy gains its strength by free chosen set of textual keywords, lack of a controlled vocabulary leads to several weaknesses like synonymy, homonymy and polysemy. For example, "mac", "apple", "macintosh" are all used to describe materials related to Apple computers. The existence of "Blog" and "Blogs" indicates that there is no collapsing of different word forms like plurals. Therefore, a set of web materials of similar content can be tagged with different keywords. In contrast with that, the same term may be used to annotate different objects. The items tagged with "engineer" varies from reverse engineering of md5 hashes to a recipe website "Cooking For Engineers". There also other problems associated with tags. After analysis of tag distribution, (Rossi, 2006) found that 40% of flickr tags and 28% of del.icio.us tags could not be found in their multilingual dictionary software. They were either misspelt, from a language not available in the dictionary, or compound words that are composed of unknown words, numbers and symbols. In addition, there are words of particular interest like "toread", "me" that reflect only users' intention of web content usage. These drawbacks hamper the performance of an information search and retrieval system incorporating such user created vocabularies. To overcome these problems, some already ongoing reseaches trying to reorganize tags according to their correlation. (Begelman et al., 2006) find clusters of strongly correlated tags by considering their co-occurrences. (Heymann and Garcia-Molina) derive a navigable hierarchical taxonomy of tags from a social tagging system using graph centrality in a similarity graph of tags. A taxonomy of Flickr tags is induced by Schmitz (Schmitz, 2006) in terms of a subsumption-based model based on co-occurrence statistics of tags. However, their works try to overcome only one weakness of tags. In this thesis, by noticing that the relations between two tags can be essentially categorized as "topic-related", various semantic relatednesses are explored in order to build a comprehensive tag ontology. The ontology is further enhanced by keywords extracted from weblog posts to improve the coverage.

1.1 Data Overview

In this thesis, "Web 2.0" application weblog and Wiktionary are the target information sources for analysis and ontology learning, which has different density of semantic relations. The following sections give an overview of the two corpora.

1.1.1 Weblog Corpus

The weblog corpus is released by ICWSM conference 2006[6], which is a complete set of weblog posts collected by Nielsen BuzzMetrics for May 2006. It consists of 14 million posts from 3 million weblogs and is given in form of 24 XML files. The collected posts are written in various languages, a list of them is given here:

Language	Percent
English	51%
Chinese	14%
Japanese	14%
Russian	6%
Spanish	3%
French	2%
Italian	2%
unknown	3%

Because we are only interested in the English posts containing tags, so we use TextCat(Cavnar and Trenkle) to filter out the posts written in other languages and collect posts with tag information. As a result, we get 58,617 annotated English posts. The majority of them are short and middle-sized documents. The distribution of post length in words is shown in figure 1.1.

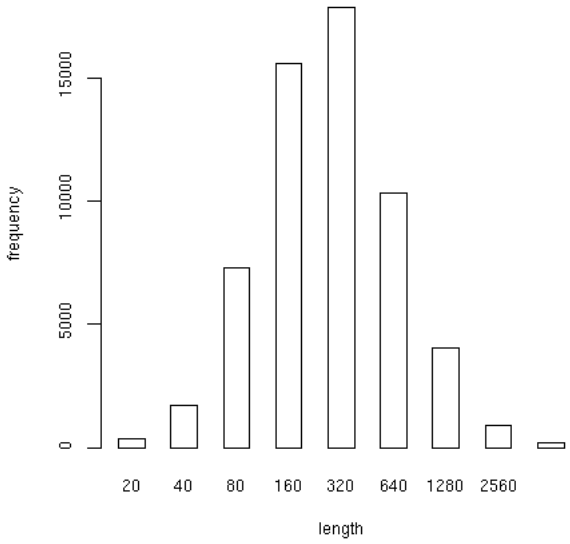

Figure 1.1: The distribution of post length in words, which is collected from all annotated English posts.

[6]http://www.icwsm.org/data.html

1.1.2 Wiktionary

Wiktionary is a multilingual online dictionary written collaborately by volunteers using wiki software. It is available in over 150 languages. There are millions of entries with definitions, etymologies, pronunciations, sample quotations, synonyms, antonyms and so on. In this thesis, we extract English noun definitions from the dump of English-language Wiktionary[7] to build a small gloss corpus. It contains 106,037 word definitions, which comprise 112,956 sentences. It worth noticing that the English Wiktionary contains about 8,400 synonyms and 1,200 antonyms but other semantic relations like hyponymy, meronymy are hardly given partly because the main online tutorial gives instruction only on how to add synonymy and antonymy. So it is a motivation of us to mine the other semantic relations from word glosses in order to overcome the weakness.

1.2 Outlines

Learning a domain-dependent ontology is our main approach to improve information search in Web 2.0. Chapter 2 covers the recent works on ontology learning from text. It shows effective methods for learning a domain dependent ontology in terms of distributional similarity and lexico-syntactic patterns. Based on that, new methods are proposed in chapter 3 to adapt the state of art methods to learn ontology from both weblog and Wiktionary. A new lexico-syntactic pattern based method is proposed to learn various semantic relations with higher performance compared to the state of art method (Snow et al., 2006). To overcome the shortage of folksonomy, a novel approach is presented in section 3.3.1 to derive a tag ontology from weblog corpus, which is enriched by our automatic tagging mechanism. The details of implementation are stated in chapter 4, which covers mainly preprocessing of data, procedural description of methods and performance issues. The evaluation of the methods and experimental results showing characteristics of Web 2.0 are presented in chapter 5. The last Chapter gives a review of the current work and an outlook of the future works.

[7] http://en.wiktionary.org

Chapter 2

Ontology Learning

"An ontology is an explicit specification of a shared conceptualization." explained Tom Gruber in (Gruber, 1993), which implies that an ontology is a formal representation of knowledge shared by a group or community. It is further suggested that ontology should be domain specific according to an application or a particular task. In this way, an ontology formalizes the intensional aspects of a domain which are not contained in a common knowledge base.

In the early time, most of the work focused on how to build a coherent, sharable and clearly defined ontology. RDF, OWL standardize expressions and formats to edit machine and human readable ontologies since the population of Semantic Web. Besides the ontologies in standard formats, there are other widely used manually built common-sense ontologies like WordNet (Miller, 1995) and Cyc[1]. As the ontologies growing large, ontology engineering becomes a painful work and the development can't catch up with the emergence of new concepts and relations. So a lot of works are carried out to build ontologies (semi-)automatically. In literatures the (semi-)automatic support of ontology development is referred as ontology learning.

In (Buitelaar et al., 2005) a layer cake model is presented to describe ontology development. As figure 2.1 shows, the building process is divided into several subtasks, which are ordered from bottom to top according to increasing complexity. Term extraction is the prerequisite of all tasks since terms are linguistic realizations of concepts. Synonym acquisition address the term variants in and between languages. The third level represents concept definition, which is one of the main concern of ontology learning. A concept has its own intensinal definition, instances and linguistic realizations, which is on Semantic Web basis. However, most works consider a concept as a collection of closely related terms as in (Lin and Pantel, 2002) due to the difficulty of automatic learning. Section 2.1 covers more details of recent works in this area.

Is-a serves as the backbone of the taxonomy hierarchy. Other relations may more or less depend on it. For instance, (Girju et al., 2003) finds patterns of part-whole relations with help of WordNet taxonomy. However, it is argurable,

[1] http://research.cyc.com

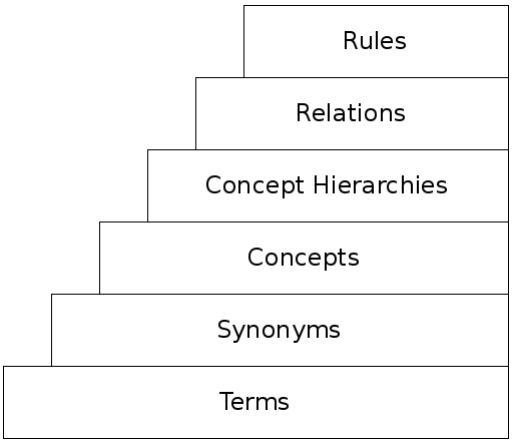

Figure 2.1: Ontology Learning Layer Cake

if it is worth to treat them separately. Let U be the set of all objects in a domain, a k-ary relation R over the sets $X_1, ..., X_k, X \subset U$ is a subset of their cartesian product, written $R \subset X_1 \times ... \times X_k$. We call r_i a instance of R, if $r_i \in R$. A function $sim(r_1, ..., r_n)$ is applied to address the degree of correspondence between relation instances. Such a function is utilized in (Turney, 2006a) to measure the semantic similarity of relation instances. If r_1 and r_2 are two 1-ary relation instances that describe an object O with properties like "O is small and" "O is big", the similarity between them is called attributional similarity, because they consider 1-ary relations as predicates taking only one argument just as attributes of the object O. When two words have a high degree of attributional similarity, they are called synonyms. If the arity of a relation is higher than one, the similarity between involved relation instances is relational similarity. For example, ("carpenter", "wood") and ("mason", "stone") are two instances sharing high relational similarity. By taking this point of view, the task of relation learning is to find relation instances of high similarity. And ontology learning becomes a problem of finding clusters of relation instances with high similarity because tasks like synonym discovery are considered as finding 1-ary relation instances of high attributional similarity. So an ontology is a tuple (U, Σ, F), where U is a set of unique terms or term sets and Σ are all relation instances in the domain. F is a set of similarity functions $sim : \Sigma \rightarrow R^+$, which assigns each relation a real number indicating the degree of similarities.

2.1 Synonym and Concept Discovery

If we view a concept as a bag of similar terms, the task of concept discovery is to find terms sharing high attributional similarity. Previous works show that the attributional similarity depends on the context of words, because semantically similar words are likely used in similar contexts, which is called distributionally similar. "beer" and "pepsi" are used frequently as the objects of "drink", so "drink [object]" provides a local semantic context to the two nouns. Hindle

(Hindle, 1990) shows that noun similarity can be effectively measured by their verb context, which consists of verbs taking nouns as their subjects or objects.

In (Lin, 1998a), the verb context is extended to allow any words that directly depend on the words in discussion. The dependency relationships[2] are identified by a broad-coverage parser PRINCIPAR (Lin, 1994). The local context of a word W is therefore a set of pairs (r,w), where w is the head or modifier of W according to the relation r. A more simple approach is introduced in (Ravichandran et al., 2005) that the local context for web corpus is restricted to two words to the left and right of each noun.

Feature Space

Despite of the different representation of the context, the co-occurrence counts of words and their local context units are taken as the basis of the features to describe each word in question. Pointwise mutual information (PMI) is calculated based on co-occurrence statistics for a better representation of data. Let $f_w(c)$ be the co-occurrence count of a local context feature f_c and a noun n_w, the PMI is defined as:

$$I_{w,c} = \frac{\frac{f_w(c)}{N}}{\sum_i \frac{f_w(i)}{N} \times \sum_j \frac{f(c)}{N}} \tag{2.1}$$

$$PMI_{w,c} = log(I_{w,c}) \tag{2.2}$$

where $N = \sum_i \sum_j f_i(j)$ if the total frequency count of all words and their features. PMI is derived from mutual information, which measures the independence of two random variables. Here PMI is a measure of the association strength between words under discussion and their local contexts. Due to the shortage of mutual information that it is biased towards infrequent words/features, a modified transformation of feature space is applied in (Lin, 1998a) by multiply $I_{w,c}$ with a factor:

$$\alpha_{w,c} = \frac{f_w(c)}{f_w(c)+1} \times \frac{min(\sum_i f_w(i), \sum_j f_j(c))}{min(\sum_i f_w(i), \sum_j f_j(c)) + 1} \tag{2.3}$$

so the modified PMI is

$$MI_{w,c} = \alpha_{w,c} \times I_{w,c} \tag{2.4}$$

Similarity Measure

Finding similar words is considered as a clustering problem. A distance function is required to measure the similarity of each example. To measure a pair of words under discussion, Hindle (Hindle, 1990) utilizes the sum of all absolute values of PMI (PMI^{abs}) as the similarity measure:

$$sim_{hindle}(w_i, w_j) = \sum_{x \in X(w_i) \bigcap X(w_j)} min(PMI^{abs}(w_i, x), PMI^{abs}(w_j, x)) \tag{2.5}$$

[2]PRINCIPAR (Lin, 1994) is based on X′ theory. A dependency relationship is a asymmetric binary relationship between a word called **head** and another word called **modifier** (Lin, 1998b). E.g. “white” modifies “cat” in the sentence “The white cat is a good cat.”

where X_w is the set of feature values of the word w. Lin proposes two similarity functions in (Lin, 1998a) and (Lin and Pantel, 2002):

$$sim_{lin}(w_i, w_j) = \frac{\sum_{x \in X(w_i) \bigcap X(w_j)} (PMI(w_i, x) + PMI(w_j, x))}{\sum_{x \in X(w_i)} PMI(w_i, x) + \sum_{x \in X(w_j)} PMI(w_j, x)} \quad (2.6)$$

$$sim_{cos}(w_i, w_j) = \frac{\sum_{x \in X(w_i) \bigcap X(w_j)} MI(w_i, x) \times MI(w_j, x)}{\sqrt{\sum_{x \in X(w_i)} MI(w_i, x)^2 \times \sum_{x \in X(w_j)} MI(w_j, x)^2}} \quad (2.7)$$

The evaluation in (Lin, 1998a) shows that sim_{lin} outperforms other 5 proposed similarity measures. sim_{lin} measures the ratio between the information shared by the objects and the information in the description of objects (Lin, 1998c).

Clustering

With proper similarity measures it is possible to calculate the distance between a set of examples. As the feature space is represented as a sparse matrix, we can make use of clustering algorithms to group similar examples.

- K-Means and Hierarchical Clustering
  K-Means and hierarchical clustering (Duda et al., 2000) are conventional and frequently used clustering algorithms. K-means assigns each example to its nearest cluster center and recompute cluster centers as the average of the cluster members. The process proceeds iteratively until the membership of each example is not changed anymore. The algorithm is linear in the number of examples but the random initialization of cluster centers leads often to poor quality of clusters and the dead-unit problem reported in (Xu et al., 1993). The class of hierarchical clustering algorithms can be further categorized as partitional, agglomerative and hybrid subcategories. The three subclasses of algorithms differ in the direction of partitioning. The partitional algorithms build a hierarchical solution by bisecting clusters repeatedly. It partitions initially all data into two clusters. According to a clustering criterion function, one of the clusters is selected and bisected until a stop criterion is met. It takes $O(nlogn)$ time for n bisections. As opposed to that, the agglomerative algorithms start from bottom to top, which try to merge similar clusters until the desired number of clusters is found. The computational complexity is at least $O(n^2logn)$ because of the calculation of the pair wise similarity between all n examples. A constrained agglomerative clustering method (Zhao et al., 2005) partitions data from both directions, which benefit from the global view of the partitional algorithms and the local view of agglomerative algorithms. A agglomerative algorithm is allowed to build a cluster tree only within one of the constrained clusters bisected by a partitional algorithm. The computational complexity is therefore $O(k((\frac{n}{k})^2 log(\frac{n}{k})) + k^2 logk)$, where k is the number of constrained clusters.

- Clustering By Committee
  All above clustering methods suffer a shortage that the number of target clusters should be known in advance. But in most cases, it is hard to

estimate. Lin proposes a new clustering algorithm CBC (Clustering By Committee) (Lin and Pantel, 2002), which is refined in (Pantel and Lin, 2002). It tries to identify some tight clusters before assigning the cluster membership to all examples. As the first step, the top k most similar words are found for each word W. By using average link clustering, which belongs to the agglomerative clustering class, the most similar words to W form a group of init clusters. By assigning a score to a cluster according to its number of members and cohesion, the most similar cluster is considered as a committee candidate. A cluster is finally identified if its similarity to previous committees is below a certain threshold. This definition of committee prefers large clusters with members having low average distances. The identification process determines that committees are the centers of each cluster. After finding the cluster centers, each example is assigned to its most similar clusters. Although the algorithm makes use of sorting und sparseness of data to reduce the time of calculating pair wise distance between sets of feature vectors, the worst case estimate is still $O(n^2k)$, where n is the number of examples and k is the number of feature.

- A Randomized Clustering Method
  CBC can find accurately similar words clusters, but it is infeasible for large scale data processing (over 10 GB). Working with large amount of data is necessary for a lot of real-world applications, especially when the useful information is quite sparse or high coverage of extracted information is required. Ravichandran proposed a randomized algorithm in (Ravichandran et al., 2005) by considering a clustering problem as finding the top n nearest neighbours. The algorithm accelerates the conventional algorithms in two steps. In the first step, a fast cosine similarity calculation is implemented by generating a signature for each vector using a locality sensitive hash function (LSH). Let u, v be two vectors from a k dimensional vector space R^k and r be a k dimensional random vector from k-dimensional Gaussian distribution, a hash function h_r is defined as:

$$h_r(u) = \begin{cases} 1 & if\ r.u \geq 0 \\ 0 & if\ r.u < 0 \end{cases} \tag{2.8}$$

  If a set of such LSHs $h = (h_{r1}, h_{r2}, ..., h_{rt})$ are applied to vector u, the signature of u is a new t-dimensional vector $[h_{r1}(u), h_{r2}(u), ..., h_{rt}(u)]$. Generally t is much smaller than k, so a vector u from R^k is represented as a t-dimensional bit vector. It is proven in (Goemans and Williamson, 1995) that

$$Pr[h(u) = h(v)] = 1 - \frac{\theta(u,v)}{\pi}$$

  where θ is the angle between the vectors u and v. The cosine similarity is rewritten as:

$$cos(\theta(u,v)) = cos((1 - Pr[h(u) = h(v)]\pi)$$

  Since $Pr[h(u) = h(v)]$ is equal to the number of different bits dividing the dimensionality of the vector, the cosine similarity is recomputed as a

measure based on the hamming distance between 2 bit vectors.

$$Pr[h(u) = h(v)] = 1 - \frac{(hamming\ distance(u, v))}{t}$$

So the algorithm benefits from dimensionality reduction and the efficiency of determing the hamming distance between 2 bit streams with preserving the cosine distance between vectors.

In the second step the algorithm groups vectors with minimal distance, which can be first considered as finding the n nearest neighbours of a given query vector q. The original algorithm utilizes a special data structure called Point Location in Equal Balls (PLEB) proposed in (Indyk and Motwani, 1998). The PLEB based algorithm finds a vector $v \in R^k$ that is a ϵ-approximate nearest neighbour of a given vector q in that for all $v' \in R^k, d(v, q) \leq (1 + \epsilon)d(v', q)$, where $d(v, u)$ measures the distance between the two vectors. However, the data structure is rather complex, so (Charikar, 2002) proposed an alternative algorithm that simplifies PLEB and preserves the same performance. If we maintain a lexicographic order of all bit vectors, similar vectors should be near each other. But some of similar vectors may still be far from the given query vector. To capture as much vectors as possible, a set of random permutations are applied to every vector, which reorder the bit values of all vectors. A permutation function $\pi : a \rightarrow b$ is a one to one mapping from the set of a to the set of b. If 0,1,2 are indexes of vector $v = [2, 4, 5]$, a possible index permutation function π is $\pi : \{0, 1, 2\} \rightarrow \{1, 2, 0\}$, after applying the function vector v becomes $[4, 5, 2]$. If a set permutation functions are generated randomly and all permuted vectors are sorted after applying every permutation, similar vectors are more likely to be arranged together. The goal of Charikar's algorithm is to find the nearest neighbours to q, but our goal is to find all vectors similar enough to q, so the algorithm is modified in (Ravichandran et al., 2005) that all closest B neighbours of q in a sorted list having hamming distance to q over a certain threshold are accepted as cluster members of the q. Overall, the algorithm takes $O(Nk + NlogN)$ time for N vectors. In case of noun clustering represented in (Ravichandran et al., 2005), $N < k$ implies that $logN << k$ and $NlogN << Nk$. Hence the time complexity is closed to $O(Nk)$, which is a huge saving from the conventional algorithms. The evaluation of (Ravichandran et al., 2005) shows that the random algorithm can retrieve 70% of the performance achieved by CBC.

The interpretation of the clustering results vary from thesauri (Hindle, 1990) to concept (Lin and Pantel, 2002). The newest proposal is from (Snow et al., 2006) that all members in a noun cluster are taxonomic cousins or coordinate terms, which subsume a common concept, because closed taxonomic cousins share similar local context and the clustering results are coarse grained compared to a manually built ontology, so that it is difficult to tell if they are synonyms, hypernyms or taxonomic siblings.

2.2 Learning the Binary Relation

This section focuses on methods to learn binary relations with high relational similarity. (Hearst, 1992) finds that some lexico-syntactic patterns like "such NP as {NP ,}*,{(or | and)} NP" can indicate a hyponymy relation between the first noun and the nouns after "as". Such patterns can be used to find relation instances with high precision. Based on the idea, a lot of works are carried out following a machine learning approach, such as (Snow et al., 2006; Suchanek et al., 2006b; Turney, 2006b)

2.2.1 Representation of Lexico-syntactic Pattern

The easiest representation is to encode a lexico-syntactic pattern as a regular expression as in (Hearst, 1992). Such kind of patterns discovery are mostly heuristic and difficult to extend. (Turney, 2006a) find phrases of interest with help of Waterloo MultiText system[3]. Then they replaces certain words of a phrase with wild cards to generate some heuristic patterns like "X nails * Y".

In (Snow et al., 2005), lexico-syntactic patterns are represented as dependency paths identified by a dependency parser Minipar (Lin, 1998b). Minipar can identify the part of speech (POS) of a word and its dependency relationships. After parsing a sentence, the relations between words and the POS of words comprise a dependency graph as the figure 2.2 shows, where the relations are the arcs connecting the POS nodes. Snow represents a relation between any two words A and B as the shortest path between the two nodes in the graph. Therefore, a dependeny path is a sequence of POS connected by their direct dependency relationship. E.g. N:mod:Prep:pcomp-n:N[4] represents the relation between "authors" and "Herrick" in the phrase "such authors as Herrick and Shakespeare". It is worth noticing that "such" is out of the shortest path though its importance. Such kind of function words is encoded as satellite links attached to one end of the shortest path. Therefore a full representation of the pattern is [such, PreDet:pre:N]N:mod:Prep:pcomp-n:N. Another problem is that "Shakespeare" should share the same dependency relationships "Herrick". However, as the parsing results of Minipar, "Shakespeare" is not directly connected to "such". By making use of he distributive nature of a conjunction relation between "Shakespeare" and "Herrick", the dependency links are populated across all conjunctions. The red arrow in figure 2.2 is a newly added link as a result of the populated dependency.

2.2.2 Snow et al.

The current most promising work about ontology learning is represented in (Snow et al., 2005) and (Snow et al., 2006). They have successfully added 10,000 new hypernyms with accuracy 80% into WordNet 2.0. To my knowledge, this is the current best result based on lexico-syntactic patterns, so we take it as the baseline method.

[3] http://multitext.uwaterloo.ca/

[4] A explanation of Minipar grammatical notations is given in appendix D

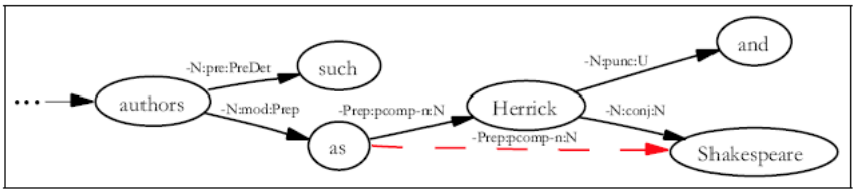

Figure 2.2: A dependency graph shown in (Snow et al., 2005), which parses a text fragment "such authors as Herrick and Shakespeare ". The above picture is taken from (Snow et al., 2005).

(Snow et al., 2006) apply a probabilistic framework that regards knowledge acquisition as a task of pattern recognition. Similar to WordNet, they define a taxonomy T as a set of pair wise relations R over a set of synsets D_T. A relation R_{ij} is an ordered or unordered pair of synsets $(i, j) \in D_T$, which encodes any binary relation like hypernymy and meronymy.

They define probability of a taxonomy as the joint probability of all possible relations.

$$P(T) = P(A \in T, B \notin T) \tag{2.9}$$

where A, B are two sets of relations that $A \subseteq T$ and $B \not\subseteq T$. Each evidence E_{ij}^R of a relation R_{ij} between two words i and j is represented as a feature vector, whose elements are the co-occurrence counts of dependency paths and the word pair. According to maximum likelihood principle, the learned model is the one best fitting the training data and maximizing $P(E \mid T)$, where E denotes the set of all evidences.

$$P(E \mid T) = \prod_{R_{ij} \in T} \frac{P(R_{ij} \in T \mid E_{ij}^R) P(E_i^R j)}{P(R_{ij} \in T)} \cdot \prod_{R_{ij} \notin T} \frac{P(R_{ij} \notin T \mid E_{ij}^R) P(E_i^R j)}{P(R_{ij} \notin T)} \tag{2.10}$$

where $P(R_{ij} \mid E_{ij}^R)$ is the conditional probability of relation R_{ij} given evidence E_{ij}^R, which is estimated by an ordinary classifier. A relation R_{ij} and its implied relation set $I(R_{ij})$ are added into a taxonomy T, if the product of the multiplicative change $\triangle_T(I(R_{ij}))$ to the conditional probability $P(E \mid T)$ is larger than 1.

$$\triangle_T(I(R_{ij})) = \prod_{R \in I(R_{ij})} \triangle_T(R) \tag{2.11}$$

$$\triangle_T(R_{ij}) = \frac{P(E \mid T')}{P(E \mid T)} = k \cdot \frac{P(R_{ij} \in T \mid E_{ij}^R)}{1 - P(R_{ij} \in T \mid E_{ij}^R)} \tag{2.12}$$

T' denotes the new taxnomony after adding a new relation R_{ij} and $k = \frac{P(R_{ij} \notin T)}{P(R_{ij} \in T)}$. They have applied their model only for the task *hypernym acquisition*. In order to improve coverage, they introduce another relation *taxonomic cousin* C_{ij}. Two synsets i and j are taxonomic cousins, if they share a parent in the hypermym hierarchy. The type of relation is found under the observation that taxonomic cousins often share similar local contexts stated in the last section. The collected evidences are used to improve the performance of lexico-syntactic

pattern based relation learning.

Another contribution of (Snow et al., 2006) is the local best search algorithm that integrates heterogeneous evidences to infer the proper relation set and solve word sense ambiguity. A new hypernym evidence (company, google) identified by a lexico-syntactic pattern may have coordinate evidences like (yahoo, google) and (yahoo, microsoft), which builds a implied relation set. The new relation and its implied relation set are added to taxonomy T when $\triangle_T(I(R_{ij})) > 1$. This is based on the fact that the right relation has always high confidence and implies related relations with high probability. The algorithm finds greedily all relation sets having multiple evidences best matching the current taxonomy. Word sense disambiguation is achieved in the same manner that right sense assignment has more evidences and higher probability than the wrong one.

2.2.3 Other Supervised Learning based Methods

- Suchanek et al.
  (Suchanek et al., 2006b) applies similar approach without taxonomy induction. They use Link Grammer Parser[5] to represent dependency paths. They name the shortest path between two words in a dependency graph a *bridge*. In addition, they take other heads and modifiers of two words in question also into account. A model is trained by both K-nearest Neighbours (KNN) and support vector machine (SVM) for comparison of pattern robustness. One of their special contribution is a special distance function of KNN, which achieves better performance than SVM.

- Girju et al.
  (Girju et al., 2003) focuses on only three lexicon-syntatic patterns to discover part-whole relation. After extraction of word pairs according to the patterns, they organize the examples into positive, negative and ambiguous examples used to learn constraints by means of decision tree and WordNet. A set of rules derived from C4.5 learner are used to discover new relations from newswire corpus.

2.2.4 Unsupervised Learning based Methods

Unsupervised learning based methods save the effort of preparing training examples and allow discovery of unknown relations. However, this approach usually has lower precision than supervised learning. (Turney, 2006b) presents an algorithm called *latent semantic analysis* (LSA) that mines large text corpora for patterns that express implicit semantic relations. It is able to discovery new patterns according to their *pertinence*, which indicates the expected relational similarity between a given word pair and prototype pairs of a specific pattern. LSA is employed in (Cederberg and Widdows, 2003) to improve precision of relations discovered by Hearst's patterns. As named entities expressing instance-of relations, KNOWITALL system (Etzioni et al., 2005) is capable of extracting facts like person and location from the Web in an unsupervised, domain-independent manner.

[5] http://www.link.cs.cmu.edu/link/

2.3 Identify Topic Related Words

It is well known that expanding query with highly related words can improve the the performance of an information retrieval system. A query can be extended with a set of semantically related words, such as their hypernym, meronyms. But some words such as "Bush", "bomb" and "Iraq", are highly related but their relations may not be defined as any conventional relations like the ones in WordNet. However, these words are the keywords describing the recent events in Iraq. So they are highly topic related and we consider the type of relation as topic related relation.

There are already plenty of works seeking to index words and documents according to their topic relatedness or cluster documents according to topic similarity. The latter issue is closely related to Topic Detection and Tracking (TDT) addressed in (Allan et al., 1998). Another interesting approach is to extract keywords from a single document which address its main concern. It allows combination of other information like position of words into the model.

2.3.1 Topic Detection and Indexing

The task of finding topic related words can benefit from the works of Topic Detection, which is an unsupervised learning task defined in (Allan et al., 1998) that recognizes if a new document falls into an existing topic or suggests a new topic. All works proposed in (Allan et al., 1998) are based on *vector space model*, so that every document is represented as a TFxIDF vector. Their works differ on the use of different clustering algrithms (hierarchical clustering or k-means) and whether to use time as an additional feature. So the results are clusters of documents sharing similar topics. (Zhao et al., 2005) follows the same approach and applies previously mentioned constrained agglomerative clustering method to group similar documents.

PLSA (Hofmann, 1999) provides a probabilistic framework to learn distribution and topic relatedness of documents and terms based on their co-occurrence frequencies. In the field of unsupervised learning it is general to consider existing samples are draw from a mixture probability distribution with unknown parameters, the basic goal is to estimate the parameters using the data. PLSA assumes documents and terms are samples drawn from a *aspect model*, which is a variant of standard finite mixture model for general co-occurrence data that associates an unobserved class variable $z \in Z = \{z_1, ..., z_k\}$, tempered Expectation Maximization is utilized to estimate the model parameters. As a consequence, a document d is not assigned to clusters but characterized by a mixture of conditional probabilities $p(z|d)$. And a word w is represented by a combination of *aspects* $p(w \mid z)$. Because of its good statistical modeling of co-occurrence, lots of research works are carried out based the model. For example, (Zhai et al., 2004) employ the linear combination of a background model and *aspect model* to study the comparative text mining problem.

2.3.2 Keyphrase Extraction

The most widely used methods are based on supervised machine learning, which classify a candidate term into a keyphrase or non-keyphrase category. (Witten et al.) start with unigrams, bigrams and trigrams after stop words removal. TFxIDF score and *first occurrence* of them, which is the distance in words of a phrase from the beginning of a document, are used as features in KEA (Witten et al.). KEA++ (Medelyan and Witten, 2006) adds the length of term in words and node degree of candidate phrases as additional features. The node degree is calculated in terms of a domain-dependent thesaurus. Since the thesaurus builds a semantic graph, the node degree is defined as the number of semantic links connected a candidate term to other candidate phrases in the same document. The evaluation in (Medelyan and Witten, 2006) shows that with the node degree feature the precision and recall of KEA++ doubles that of KEA. (Hulth, 2003) utilizes TF and IDF as two separate features instead of one. In addition, they show that empirically edited POS tag patterns of candidate terms such as *NOUN NOUN* and *ADJECTIVE NOUN* are effective in identifying important phrases. They achieves the highest precision (41.5%) and recall (46.9%) reported in (Hulth and Megyesi, 2006).

There are also other approaches for identifying topic words. In (Clifton et al., 2004), a collection of named entities are used to represent the topic of a document. By applying a modified association rule learning, they identify the frequent co-occurred entity sets (frequent itemsets) and group them together to build clusters sharing similar topics. Each document is then assigned to each cluster according to their TFxIDF distance to cluster centroid. To avoid too fine-grained segmented clusters, the candidate clusters are merged together according to the similarity of their document members. TextRank algorithm (Mihalcea and Tarau) represents each document as a text graph, whose vertices are terms connected according to their co-coorrrence relation. All terms are then ranked similar to Google's PageRank algorithm. Final keyphrases are chosen after post-processing the top N terms.

Chapter 3

Learning Ontology from Web 2.0

Learning ontology from Web 2.0 digital assets can't simply apply the methods by learning from text. The grassroots writers show inconsistent style of writing and the way of expression varies from people to people. Although writers from Non-English speaking countries contribute lots of content, they make also plenty of errors while writing. The existence of splog and duplicate documents make it even more difficult to extract expected information from the noisy data. Additional preprocessing steps have to be carried out in order to get more normalized information, which will be mainly covered in the next chapter. This chapter concentrates on algorithmic improvement and explanation of new methods. Relation instances of different arities are discovered in terms of distributional similarity, lexico-syntactic patterns and topic relatedness. Folksonomy existing in the weblog corpus is another focus of this chapter. In section 3.3.1 a tag ontology is derived by exploring the semantic relatedness between tags.

3.1 Learning Coordinate terms from Weblog Corpus

Coordinate terms are considered as 1-ary relations with high distributional similarity. Besides high precision the algorithm should be computationally efficient enough in order to process large amount of data. The randomized clustering method proposed in (Ravichandran et al., 2005) is the most interesting candidate because of its efficiency and sufficient accuracy. However, practice shows that direct implementation of the algorithm can easily run out of memory if the grammatical context first proposed in (Lin and Pantel, 2002) is applied. A feature set as large as 250,000 can easily consume 2 GB memory for 1000 random vectors. One solution suggested in (Ravichandran et al., 2005) is to use a simple context definition that comprises two words to the left and right of a noun. But it will lose useful information like the verbs as head of a noun. One solution is to write the random vectors into disk and sequentially read them during hashing. However, in practice it takes a much longer time than keeping

all random vectors in memory. Our solution is based on the observation that there are a lot words occur rarely in weblog corpus, which contribute few to context building. If we allow only the words having sufficient occurrence across several documents, the dimensionality can be strongly reduced. E.g. there are 335,584 unique terms identified by Minipar from 52,893 blog posts, if we select the words having normalized *idf* ranging from 0 to 0.9, the number of terms are reduced to 29,098. Another preprocessing step is to convert the numbers in a short term to a unique label and append the rest of characters to the end of it. In this way "10:30" is converted to "DIGIT:" and "20" is "DIGIT". So the various terms containing numbers are generalized into a few fixed patterns.

Besides additional preprocessing, the generation of random permutations should be improved. In (Ravichandran et al., 2005), a pseudo random number generator is utilized to approximate random permutations:

$$\pi(x) = (ax + b) mod\ p$$

where p is prime and $0 < a < p, 0 \leq b < p$. a and b are chosen randomly. If p is the size of a bit vector, a index x can be projected randomly to any other index of the vector. So the generator can not guarantee a one to one mapping, there is a probability that the different indexes are mapped to the same one. The alternative is a random k-permutation algorithm[1]:

```
function kpermutation(ar, k)
{
  for (i=1 to n) ar[i] = i;
  for (i=1 to k) swap(ar[i], ar[Random(i,n)]);
}
```

Random(i,n) is a random generator selecting a random number between i and n. The algorithm swaps k elements of n-dimensinal array **ar** without repeated index mapping.

The last step of the original algorithm is to find a set of vectors as nearest neighbours of a given vector q in a sorted list. After fast hamming distance search, several duplicate or quite similar clusters will be generated since the vectors belonging to a cluster are nearest neighbours to each other. If we consider each cluster as a radius-r ball centered at v (equal ball), there is a simple solution to merge similar clusters together. At first all equal balls are sorted by size in descending order. Every time we pick one cluster from the sorted list and calculate overlapping rate of other equal balls centered at all its members. If the overlapping rate of two equal balls is above a threshold θ_2, they are merged and both are removed from the sorted list. Here pair wise similarity calculation is abandoned because its computational complexity is $O(N^2)$, which is higher than the overall algorithm. The modified algorithm is reformulated as follows:

1. Given n vectors in k dimensinal space R^k, choose d ($d \ll k$) k-dimensional random vectors $(r_1, r_2, ..., r_d)$. Each element in r_x is drawn from a Gaussian distribution with mean 0 and variance 1.

[1] http://www.techuser.net/randpermgen.html

2. For every vector $v \in R^k$, a set of LSH $h_r(v)$ (equation 2.8) generates the signature of v as:$u = \{h_{r1}(v), h_{r2}(v), ..., h_{rd}(v)\}$.

3. Generate randomly a permutation function π by kpermutation function. We sort lexicographically n permuted vectors.

4. For each vector v in the sorted list, we look up B vectors upwards and B downwards. If the hamming distance of a vector to v is below a threshold θ, the vector is stored into a cluster centered at v. After finding all nearest neighbours, it goes back to the third step, until all q permutations are done. This is a small modification of the original algorithm, which generates all permutations all at once before searching. It saves the storage of all permutation functions.

5. Sort all initial clusters by size in descending order. Select a initial cluster A in sequence, calculate the overlapping rate for all clusters centered at its members. If the overlapping rate is above θ_2, the cluster B centered at that member is removed from sorted list and all its members are added into A.

With additional preprocessing steps and algorithmic improvement, the clustering algorithm is capable to process efficiently large amount of data on a common workstation PC.

3.2 Learning Relations from Lexico-syntactic Patterns

As stated in the section 2.2, lexico-syntactic patterns are widely used to recognize word relations from free texts. The baseline method from (Snow et al., 2006) is the first choice to us to discover semantic relations from the text. However, the direct implementation of the baseline method does not fit into memory and shows poor performance on the annotated English posts[2]. We use a new data structure *dependency tree* to make the original algorithm memory efficient and be able to incorporate context information to capture more patterns.

3.2.1 Dependency Tree

In (Snow et al., 2006), each dependency path is viewed as a standalone feature. Such a path is mainly the shortest path between two words under discussion denoted as $POS_{head} : r_1[POS : r]* : POS_{tail}$. For simplicity, we refer in latter part the shortest path between two words as a *bridge*, which is first used in (Suchanek et al., 2006b). The satellite links $[r, POS_{\{w1,w2,...\}}]*$ are a set of words of the same POS that directly depend on either end of a *bridge* with relation r. So a complete dependency path is of form $[r, POS_{\{w1,w2,...\}}] * POS_{head} : r_1[POS : r]* : POS_{tail}[r, POS_{\{w1,w2,...\}}]*$. If we consider a POS as a node and r is an arc connecting them, all dependency paths comprise a tree (figure 3.1), where the root is POS_{head} and a POS_{tail} is a special nodes called tail nodes locating at the bottom of a tree. Because a *bridge* can start from either side, we

[2]Experimental results are given in section 5.2

can let a dependency path always start from the word locates on the left side of the other.

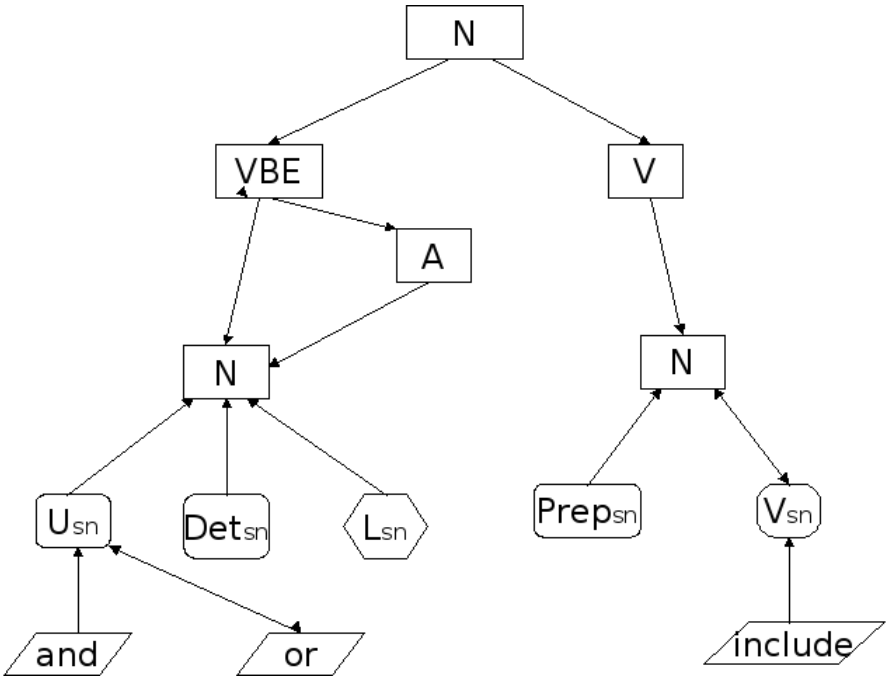

Figure 3.1: A dependency tree. Each rectangle represents a POS node in a shortest path. All POS_{sn} are POS nodes of semantic neighbours. Word nodes take the form of parallelograms. Entity nodes are represented as hexagons. Each arrow is the dependency relation connecting the nodes.

The satellite links in (Snow et al., 2005) are some heuristic patterns attached to either end of a *bridge*. A link ignoring function words is again a (POS, R) pair, which has the same form as a node in a path. So it can be attached to a tail node it depends on. In order to distinct to which word under discussion it has dependency relationship, the kind of nodes are denoted as $SN_{pos}^{H|T}$, where H or T means head or tail end of a *bridge*. The function words of a $SN_{pos}^{H|T}$ are considered as a set of word nodes (WN) as its children. Named entity class of a word is also an important part of a pattern. It is in form of a entity node $entity_p$ connecting to a $SN_{pos}^{H|T}$ or a tail node, which depends on if it takes place outside or inside a *bridge*. If it occurs in a *bridge*, the p is used to denote the position information. All $SN_{pos}^{H|T}$,WN and $entity_p$ comprise a special collection of nodes, which provides context information in addition to *bridges*. Such kind of nodes are referred as *semantic neighbours* (SN) in the following part.

If we consider a dependency path as a random walk starts from POS_{head} and ends at POS_{tail}, we can store the access count C_A in each node to get a state transition matrix. Given a parent node, we can estimate the occurring probability according to C_A of its children. That is why we call it "dependency tree", since parent and child nodes are semantically dependent and the occurrence of

a child node depends on its parent nodes according to the current modeling. Furthermore, if we want to observe the occurrence of a certain relation R, the frequency information C_R can also be stored in each tree node. Together with the same information stored in SNs it is possible to estimate how likely a path is correlated to R.

We still need to store the co-occurrence information between pairs of nouns and dependency paths as in (Snow et al., 2005, 2006). In our tree modeling, a subtree taking a tail node as its root is stored as the corresponding position of a feature vector, since it represents a dependency path. All such subtree vectors comprise the instance base of a dependency tree.

3.2.2 Baysian Network Estimation

We can directly estimate how likely a relation R occurs in a dependency path P from a dependency tree. Given a tree node n_a other than SN nodes, we obtain the probability that a relation R occurs in a dependency path starting from POS_{head} and ending at n_a as:

$$P(n_a) = \frac{C_R^{n_a}}{C_A^{n_a}} \tag{3.1}$$

But the situation will become complex when wir take SN nodes into account. A *bridge* can have different number of SN nodes, which can co-occur in a dependency path. WN depends strongly on $SN_{pos}^{H|T}$ and we have to remove SN nodes with low occurrence to obtain stable patterns. Following a statistical way we consider each node as a Bernoulli variable x with range $\chi = \{1, 0\}$. For a SN node the two values indicates if it occurs or not and for a tail node 1 represents the existence of relation R. Because of the computational complexity considering the whole tree, we focus only on a subtree with a tail node as root. The joint probability of all nodes M having its root at a tail node N (figure 3.2) is:

$$P(M) = \prod_{x_i \in M} P(x_i \mid ancestor(x_i)) \tag{3.2}$$

where M are all nodes in the subtree including the tail node N and all its SN nodes, $ancestor(x_i)$ denotes all ancestors of node x_i. Compared to figure 3.1, there are additional links from word nodes to the tail node to show their dependency. If it is assumed that each node depends only on its parent and is independent of the other nodes, we have:

$$P(x_i \mid ancestor(x_i)) = P(x_i \mid parent(x_i)) \tag{3.3}$$

where $parent(x_i)$ is the direct parent nodes of x_i. Combining equation 3.2 and 3.3, we obtain

$$P(M) = \prod_{x_i \in M} P(x_i \mid parent(x_i)) \tag{3.4}$$

If we assume the condition holds also for the other nodes, a dependency tree becomes a Baysian Network, in which an edge leads from node a to b means that the variable a depends on b. So our prior knowledge about dependency paths is

represented by the network structure. Currently we inspect only a subtree with a tail node as root, so tail nodes are assumed to have no parent node.

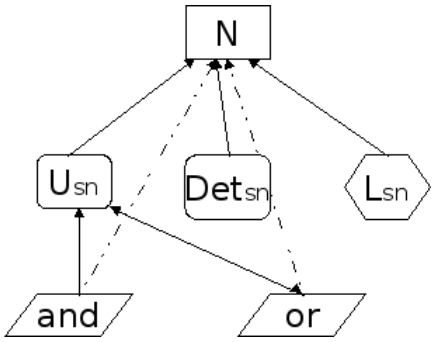

Figure 3.2: A dependency subtree. Each rectangle represents a POS node in a shortest path. All POS_{sn} are POS nodes of semantic neighbours. Word nodes take the form of parallelograms. Entity nodes are represented as hexagons. Each arrow is the dependency relation connecting the nodes. Dashed arrows are newly added relations

To obtain the likelihood of a dependency path indicative a relation R, let N be the class attribute and all SN nodes be binary features, the probability is given as:

$$P(N = 1 \mid SN) = \frac{P(N = 1, SN)}{P(N = 1, SN) + P(N = 0, SN)} \tag{3.5}$$

where SN is a collection of all SN nodes in the subtree. From above equations it is easy to see that we obtain $P(N = 1 \mid SN)$ easily if $P(x = 1 \mid parent(x))$ is known, since $p(N = i, SN) = P(N = i) \prod_{sn \in SN} P(sn = 1 | parent(sn))$. The probability of a word node and a SN_{pos} node can be derived directly from their frequency count and that of their parent nodes:

$$\begin{aligned}
P_{wn}(x = 1 \mid SN_{pos} = 1, N = 1) &= \frac{C_R^{wn}}{C_R^{SN_{pos}}} \\
P_{wn}(x = 0 \mid SN_{pos} = 1, N = 0) &= \frac{C_A^{wn} - C_R^{wn}}{C_A^{SN_{pos}} - C_R^{SN_{pos}}} \\
P_{SN_{pos}}(x = 1 \mid N = 1) &= \frac{C_R^{SN_{pos}}}{C_R^N} \\
P_{SN_{pos}}(x = 0 \mid N = 0) &= \frac{C_A^{SN_{pos}} - C_R^{SN_{pos}}}{C_A^N - C_R^N}
\end{aligned} \tag{3.6}$$

We consider only presence of SN nodes because absence of a SN node is modeled as missing value. It is based on the consideration that "such" is not used to describe the relation between two hypernyms just because the number of examples are small. Another problem is zero frequency when $C_R = 0$ or $C_R = C_A$, so that the probability of a node is zero. No matter how high the other variables are, the joint probability is always zero. The problem can be solved by *Laplace estimator*, which inits every count by 0.5. The corresponding probability of a node x is:

$$P_x(x = 1 \mid parent(x)) = \frac{C_R^x + 0.5}{C_R^{parent(x)} + 1} \tag{3.7}$$

$$P_x(x = 0 \mid parent(x)) = \frac{C_A^x - C_R^x + 0.5}{C_A^{parent(x)} - C_R^{parent(x)} + 1} \tag{3.8}$$

The last step is to calculate the likelihood of a relation given a prediction probability vector, which contains the estimated likelihood of all subtrees associating a word pair. One simple method is to choose the max value of them because a positive example may exist in different dependency paths indicating the strength of relation differently. The strongest pattern determines the association strength of two terms. It can show the performance of a single dependency path but can not process noisy data properly. Another way is to use an ordinary classifier to learn the distribution based on BNE predictions, which can be viewed as feature space being transformed into a "confidence space".

3.3 Learning Topic Related Relation

In a corpus like weblog, people write down their feeling and opinions about daily events or their personal lives. The way of writing is casual, emotional and there are only a little expressions reflecting semantic relations between words. Since the majority of words are related because they are used to discuss certain topics, it is of our interest to find the topic related words from weblog corpus and consider it as a task of finding topic related relations. A topic related relation is a n-ary relation with $n > 1$. A group of such relation instances with high similarity are likely used to express a similar topic. Because the tags in a post are likely to describe its main theme, in section 3.3.1 a tag ontology is derived from folksonomy of weblog corpus, which is further extended with our keyphrase extraction method proposed in section 3.3.2.

3.3.1 Tag Ontology in Weblog

As mentioned before, folksonomy shows a new way of organizing large amount of information by collaborately annotating and categorizing content. In weblog corpus, we find that the tagging service of weblog is either provided by existing blog systems or tagging service providers like Technorati[3]. A typical tag in a blog post is a link with a special value "tag" of the attribute *rel*. The other representation of tags are some word lists occurring after "TAGS:" at the end of a post. From all 5 million English posts we extracted 52,892 unique posts annotated by 55,492 unique keyphrases after removing near duplicate ones.

The majority of tags are from tagging service providers. Among them Technorati is the undisputed market leader since 64% tags come from their service. Politics, blog, music, media, news are the top frequent used keywords, which still stay in the top 100 list on the front page of technorati.com. Another interesting aspect is that users intend to choose keywords of a general category like music, politics to describe content, even though they do not occur in the text. Only 2220 keyphrases are found in their annotated texts. 71% posts do not contain any phrases that are used as their tags[4].

[3]http://www.technorati.com

[4]Both post texts and tags are stemmed by Snowball before checking if a tag is contained in a text.

cluster name	cluster members
atomic kitten	natasha hamilton,atomic kitten
superman	superman return torrent,superman vs jesus, superman movie,supperman return
fusion energy	nuclear fusion,fusion energy

Table 3.1: Three small example clusters found according to context similarity. Every row of the table represents a distinct cluster and cluster members are separated by comma. The cluster names are chosen only for reading purpose.

As discussed in the first chapter, despite of obvious advantages of a social classification system, people find its flaws during the daily usage. One drawback lies in that uncontrolled vocabulary allows existence of different word forms. This issue can be well solved by stemming technique. Other than rule based suffix stripping algorithms like porter algorithm (Porter, 1997), which often produces unsatisfied results, we employ the lemmatization[5] method of tree tagger (Schmid, 1994) to attain higher accuracy. The lexicon of Tree tagger consists of three parts: a *fullform lexicon*, a *suffix lexicon* and *default entry*. The *fullform lexicon* is created from Penn Treebank corpus, which contains roughly 2 million words. A word is first looked up in the lexicon in its original form and then searched again in lowercase if the fullform is not found. If it fails again, suffix stripping is utilized by finding the most likely suffix from the suffix lexicon. The suffix lexicon is built automatically from the same corpus in similar manner as building a decision tree. The *default entry* is returned only if the search in the suffix lexicon fails. So the approach benefits from both lexicon based stemming and probabilistic inference.

We solve sense ambiguity by clustering similar tags according to their associated documents. Other than (Heymann and Garcia-Molina), which uses the whole document as a unit of context, we decompose the referred posts into a bag of words. Tags are aggregated into tag vectors. The number of posts that a term co-occurs with a tag serves as an element of the tag vector. This results in a sparse matrix that represents tags. The values of the matrix is further transformed by calculating the pointwise mutual information of each entry.

$$PMI(t,c) = log \frac{\frac{f_t(c)}{N}}{\sum_i \frac{f_t(i)}{N} \times \sum_j \frac{f(c)}{N}} \tag{3.9}$$

where $f_t(c)$ is the number of annotated documents by a tag t and a context term c, $N = \sum_i \sum_j f_i(j)$. The modified randomized clustering algorithm proposed in section 3.1 is employed to find groups of highly related tags. As a result, we found 3546 unique clusters from all 58,617 tagged posts. Table 3.3.1 presents some example clusters found in the corpus[6].

In order to facilitate browsing of tags, (Heymann and Garcia-Molina) and (Schmitz, 2006) seek different ways to organize tags into a hierarchy. (Heymann

[5]Because of the lack of context, the lemmatization method is used to solve a stemming problem.

[6]30 exmaple clusters are presented in appendix C

and Garcia-Molina) builds a *tag similarity graph* where each tag is represented as a vertex, and two vertices are connected by a edge if they both annotate a given object above a certain times. According to their algorithm, a parent vertex is identified if it has high graph centrality. (Schmitz, 2006) utilizes a simple model addressing subsumption relation between tags. A tag Y is a parent of another tag X if $P(Y \mid X) \geq t$ and $P(X \mid Y) < t$, where t is a user specified threshold. Both X and Y should occur more than a minimal times.

The above two methods capture a pattern that users like to annotate a object with several keywords of different specialization levels. For example, a user likely to annotate his article about jazz with two keywords "jazz" and "music". And "music" is reused more often than "jazz". The idea is similar to find association rules from a set of co-occurred objects. An association rule is defined in (Agrawal et al., 1993) as follows:

> Let $I = \{i_1, i_2, ..., i_k\}$ be a set of attributes called items and $T = \{t_1, t_2, ..., t_m\}$ be a set of transactions, where $t_i, i \in [1, m]$ is a subset of I, an association rule is an implication of the form $X \Rightarrow Y$, where $X, Y \subset T$ and $X \bigcap Y = \emptyset$.

The intuitive meaning of such a rule is that transactions contains items in X tend to also contain the items in Y. *Support* and *confidence* proposed in (Agrawal et al., 1993) are the first measures for mining such rules. The *support* of a rule is the joint probability $P(X, Y)$ of two itemsets, which describes the co-occurrence of X and Y. The *confidence* of the rule $X \Rightarrow Y$ is the conditional probabillity $P(Y|X)$ that transactions contains Y, given that they contains X. (Brin et al., 1997) uses *conviction* instead of *confidence* because $P(X, Y)/P(X)$ could equal $P(Y)$, which means the occurrence of Y is unrelated to X and the measure is still high enough to hold the rule. They define *conviction* as $P(X)P(\neg Y)/P(X, \neg Y)$ under the consideration that it includes $P(Y)$ and $X \Rightarrow Y$ can be rewritten as $\neg(X \wedge \neg Y)$ so that *conviction* measures how far $X \wedge \neg Y$ deviates from independence. We use both *support* and *conviction* to find rules like "user tags an object with a word X also tends to annotate it with another word Y". We do not use the subsumption concept because although some rules like $jazz \Rightarrow music$ represent semantic is-a relation, there are also a lot of cases that the rules interpret other types of semantic relatedness. For example, $protest \Rightarrow immigration$ hold true because *protest* is used to refer the same topic together with *immigration* and *immigration* is a more general category than *protest*. We say that a set of such rules build a hierarchical like structure if an association rule is regarded as an edge connecting two term sets. Since association rules are directional, the more general categories such as "politics" and "music" are placed on top of the structure. Additionally, we define another relation $equality^{\epsilon}$ as

$$equality^{\epsilon}(A) = \frac{P(A_1, A_2, ..., A_k)}{MAX(P(A_i))}, i \in [1, k] \tag{3.10}$$

since $X \Rightarrow Y$ and $Y \Rightarrow X$ implies $X = Y$. If two such rules have *confidence* above a common threshold ϵ, we call them $equality^{\epsilon}$. It makes sense because if $P(A_1, A_2, ..., A_k)/MAX(P(A_i)), i \in [1, k]$ holds true so as any $P(A_1, A_2, ..., A_k)/P(A_i), i \in [1, k]$. The use of *confidence* does not cause the previously mentioned problem

because the related tag sets have similar prior probability if the rule holds. As a result, the tag ontology is a graph consisting of three types of relations.

3.3.2 Keyphrase Extraction

Previous analysis on weblog data shows that 71% annotated posts do not contain any phrases that are used as their tags and in our weblog corpus, less than 4% English posts are annotated. If we compensate current tags with keywords extracted directly from weblog content, we can extend the power of folksonomy to any size of data.

PLSA provides good stochastic representation of documents and terms related certain topics. However, the iterative nature and high computational complexity of EM makes the processing of large amount of data infeasible, especially when the number of clusters is large. Even for a small amount of weblog data, the determination of the number of topics is quite arbitrary, which can lead to unsatisfied results in practice. The research on keyphrase extraction provides a good alternative. It allows high speed of data processing and the combination of information in addition to co-occurrence. If a document is represented as a set of keywords, the keywords can be viewed as being automatically annotated, so that it owns the same nature as tags. Following a supervised learning approach, we separate the extraction process into two steps, candidate phrases identification and keywords filtering.

Candidate phrases are all unigrams, bigrams, trigrams having predefined POS patterns, because they cover already over 90% of tags (figure 3.3) and long n-grams in blogsphere intend to be arbitrary sentences or misused phrases. As in (Hulth, 2003) only nouns and noun phrases are considered, since they comprise the vast majority of tags and the involvement of other types of phrases leads only to reduction of performance. The idea is similar to (Hulth, 2003) despite we use the different parser to identify NP-chunks.

The filtering process utilizes a learned model to identify the most important words among all selected n-grams based on a set of features. Each candidate term is represented by the following nine features:

- **TFxIDF** value of a term, which is also used in both KEA and KEA++.
- **Relative position of the first occurrence**, which is defined as *sentence index / number of sentences*. In our data, it performs better than the relative position in words.
- **The length of candidate terms in words**, which is a simple numerical feature.
- **Linked words** is a binary property whether a candidate terms is used in a hyperlink.
- **Emphasized words** is also a binary feature whether a candidate term is emphasized by any html markups like **I** (italics) and **B** (boldface).

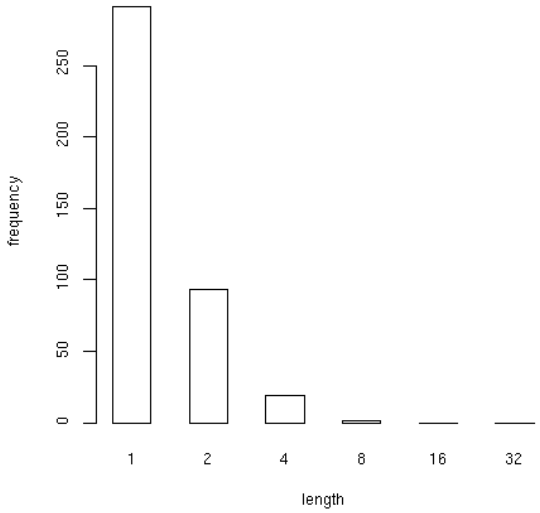

Figure 3.3: The histogram of tag length in words sampled from weblog corpus. Only tags reoccurred in their associated posts are taken into account.

- **Named Entity class** of a candidate term is represented as three binary features to determine if it is a person, location or an organization.
- **Node degree**, which shares the idea of KEA++ that measures the semantic relatedness to the other candidate phrases in the same document. The learned tag ontology is our domain specific thesaurus, which builds a semantic network of topic related terms. Rather than count the number of semantic links to the other candidate terms, we define the node degree as the sum of edge weight to the other candidate terms. Since the ontology consists of relations built upon context similarity and association rules, the edge weights are treated separately. If two terms comprises a relation instance according to context similarity, the weight of arc is only included when it is over a certain threshold θ, because experiments show that low strength of relation produces more noise than useful information. If two terms are found co-occurred above a minimal support, their *interest* measure is used as the weight of the arc. The interest measure between two terms A and B are defined as follows:

$$interest(A, B) = \frac{P(A, B)}{P(A) * P(B)} \tag{3.11}$$

  conviction is not chosen here because *interest* is symmetric so that achieves higher coverage of relations.

However, the significant PoS pattern feature in (Hulth, 2003) has only negative effect in our cases because the vast majority of tags are single terms whereas the most keyphrases in (Hulth, 2003) are multiword phrases.

In the last step, any classifier supporting both numerical and nominal attributes can be used to learn the model. The logistic regression classifier from Autonlab is chosen because it yields good results and is computationally fast. To select keywords from a new document, the learned model is applied to the feature vectors of candidate terms extracted from the document. Top ranked candidate terms are selected as the final set of keyphrases.

Chapter 4

Implementation Details

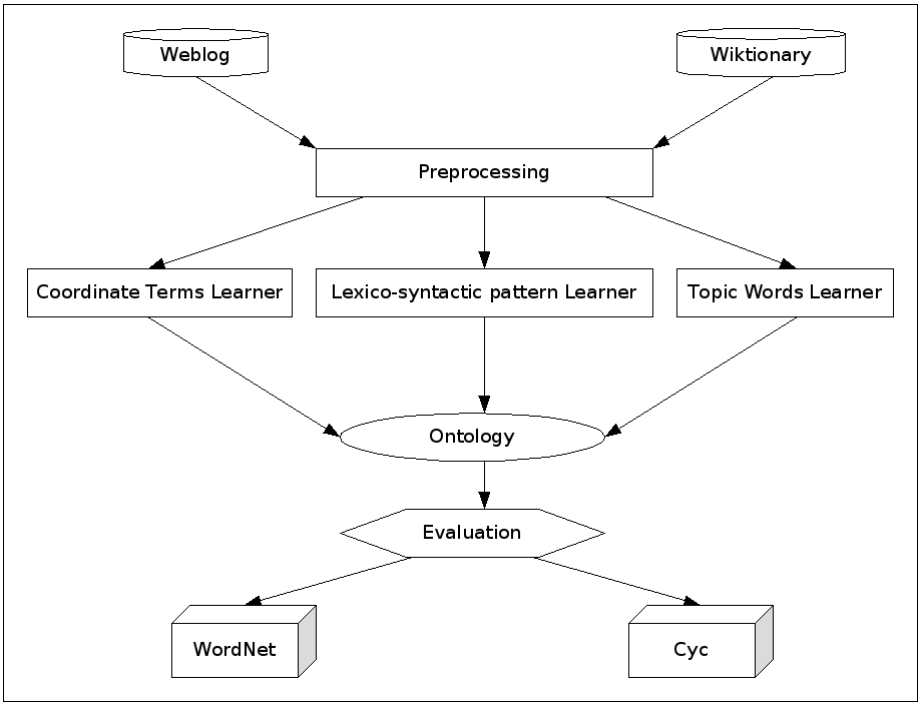

Figure 4.1: Components overview

In previous chapters, different methods are proposed to learn ontologies from weblog corpus and Wiktionary glosses. They are the core of our ontology learning system. Figure 4.1 gives an overview of all the components. The whole system works like a workflow. The text data flow through the preprocessing component, in which text data is normalized for further building of feature spaces required by relation learners. Coordinate terms learner extracts the taxonomic cousins according to distributional similarity. The lexico-syntactic pattern based learner has implemented the methods in (Snow et al., 2006) and

Bayesian network learners. The tag ontology is learned by Topic words learner, which contains also the tagging recommendation component.

Since the weblog corpus is large in size, the system should be capable of processing large amount of data. And some experiments are long running programs, which needs fast retrieval and store of data efficiently. Mysql relational database is chosen as the main data backend because its reliability and high performance. Since the system is written in java, it is fast and convenient to use Berkeley DB to serialize tree models and processing results into disk. In this system all important temp results are serialized as "save points" in order to avoid rerunning of some long-time experiments like parsing and also facilitate fast prototyping. Most of functional components represented here do their task not other than reading data, processing them and writing the results into database. So they are generalized as data processors, which comprise the "steps" of a workflow. Logging component traces errors and important steps during data processing so that debugging of long running programs and review of experiments are possible.

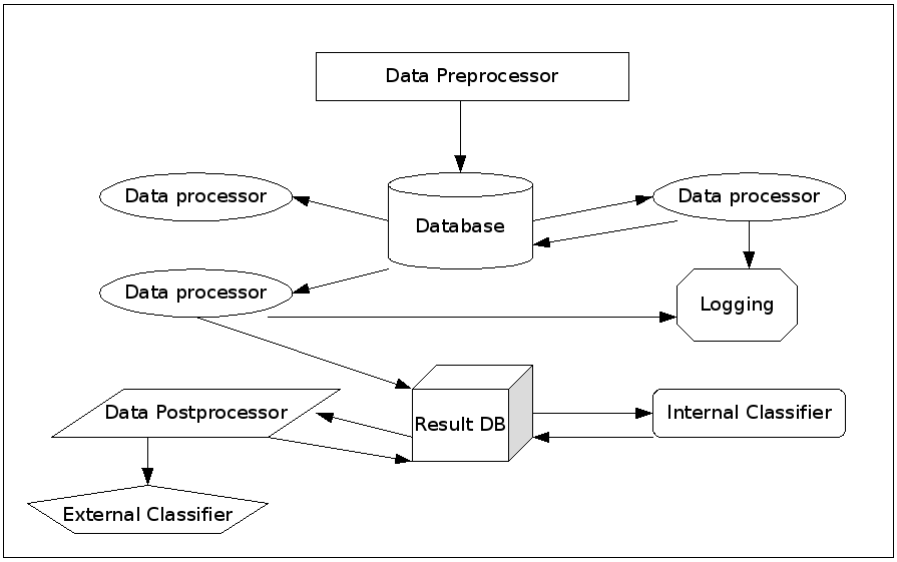

Figure 4.2: System Architecture. Arrows denote the direction of data flow. The geometrical forms will be reused in the following sections to denote the same class of components.

4.1 Data Preprocessing

Data preprocessing is the first step that normalizes the real-world data before they are fed into a classifier. According to experience, this step takes mostly 80% time of the entire work in a typical pattern recognition task and is one of the most important factors influencing the end results.

Weblog and Wiktionary are the main data sources under discussion in this thesis. Although they share some common characteristics of web 2.0, their differences determine the different preprocessing steps.

4.1.1 Weblog corpus

Each weblog post is a normal HTML page with some meta data like permalink and post time. They are written in different languages and contain occasionally the category information "tags". The preprocessing pipeline is demonstrated in figure 4.3.

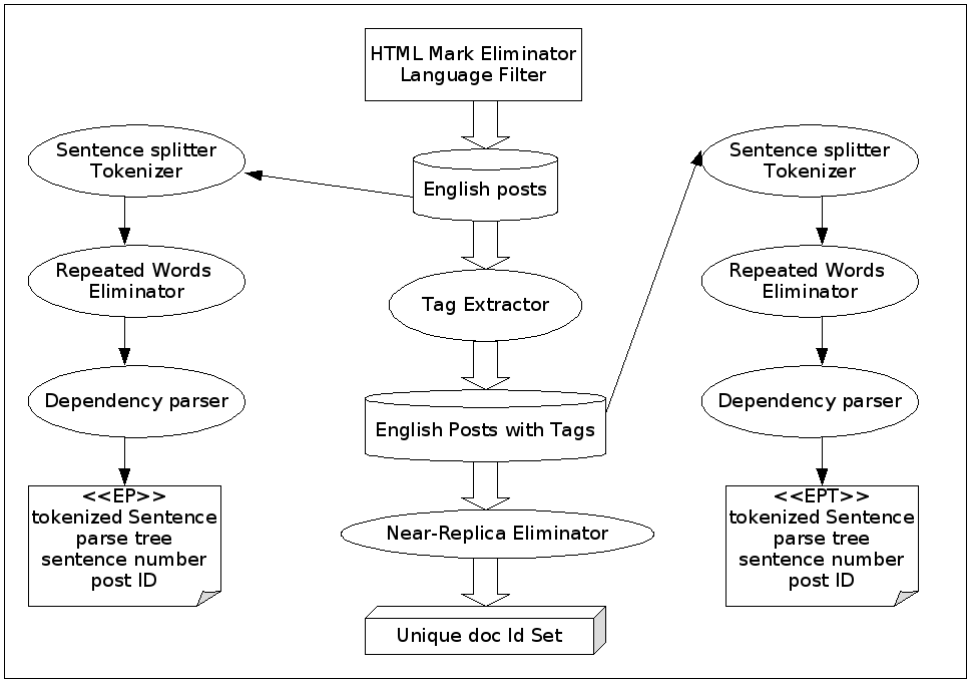

Figure 4.3: The preprocessing workflow of weblog corpus. EP and EPT denote the corresponding results.

- **HTML eliminator and language detection** the HTML pages are first converted into plain text so that it is possible to use TextCat to make language detection. All English posts are then stored into a table of database together with the meta data. It took a week long to finish processing all 24 XML files.
- **Tag extraction** The domain of interest is the English posts with blog tag information. Two types of blog tags are extracted from HTML pages. One type of tag is wrapped by a hyperlink mark having a special attribute rel="tag". This type of tags are mostly provided by a tag service provider like Technorati. The other type of tags are some key phrases written simply behind the word "TAGS" and locate mostly at the end of a post. After extracting the two forms of tags we get 58617 English posts.

```
<a href="..." rel="tag">movie marketing</a>
```

- **Sentence splitter and tokenization** The tokenization process integrates NER detection so that every named entities are recognized as a single token. We use Standford named entity recognizer (Finkel et al., 2005) to detect the words and sentence boundaries as well as NER detection. Each tokenized sentence is stored in form of XML into a sentence table having a foreign key pointing back to the corresponding post.

- **Repeated words elimination** In blogsphere bloggers intend to repeat some words like "so so so big" to emphasize their opinions or express their feeling. Minipar is trained in corpus SUSANNE, so it is incapable to recognize such kind of phrases properly. Experiments show that if a word is repeated too much times, Minipar will consume all memory just for parsing a sentence. The solution is simply merge the identical tokens into one in a sentence. As a result, 577703 sentences are found to have repeated words and among them 127512 tokens are repeated more than 2 times.

- **Duplicate and near duplicate documents elimination** Duplicate documents and splog have strong negative effect on the classification performance. The signature-based near-replica detection method proposed in (Kolcz et al., 2004) is applied to remove all duplicate and near-duplicate posts. The original algorithm is inefficient in memory management, so a modified procedure is given as follows:

  1. Collect first the *idf* of stemmed terms from a given corpus. A lexicon L is generated by imposing an upper and lower limit on the *idf* for words, since terms with mid-range *idf* values are shown to be more useful in duplicate detection (Chowdhury et al., 2002).
  2. We go through each document. A document is represented as a set of unique terms. For a current document, the unique term set is modified by randomly adding and removing a term from L, with m such changes in total. Generate a secondary lexicon by randomly removing a fraction p of terms from L. The intersection between the modified term set and the secondary lexicon is created and sorted in lexicographic order. Like I-Match (Chowdhury et al., 2002), a signature is generated for each sorted term set by adding each term to the SHA1 (NIST 1995) digest. The document ID is stored in a hash table whose keys are signatures.
  3. (Kolcz et al., 2004) shows that performance can be improved by means of several secondary lexicons. If we repeat the last step K times, every document is represented as a tuple of signatures ($digest_L$, $digest_{L1}$,...,$digest_{LK}$). Two documents are regarded as similar if they share at least one signature. However, if we keeps all hash tables in memory, it is impossible to process millions documents. So we consider each document as a node of a graph. Two documents are connected if they share at least a signature. It is a small improvement over the original algorithm, which generates K secondary lexicons and $K + 1$ signatures all at once.

The original algorithm considers each entry in the hash table as a cluster of near duplicate documents. In practice, it is found out that there are some similar clusters that should be merged together, because we ignore the cases that if document A and B share a signature S_a , C and B are recognized as similar by another signature S_b, so that C is not included into the cluster of A and B. (Kolcz et al., 2004) does not show how to merge them together. The brute-force method is to compare each entry with other map entries to check if they have at least one element in common. It will take in worst case $O(N!)$ for N entries. Since we represents all documents as a set of graphs, a cluster of similar documents can be found by any greedy graph search. The time complexity is $O(n)$ for a graph of n nodes.

- **Dependency parsing** Dependency parsing is accomplished by Minipar, a fast principle-based English parser. The parser reads tokens of each sentence from sentence table and stores the parsing result also in XML back into the sentence table.

4.1.2 Wiktionary

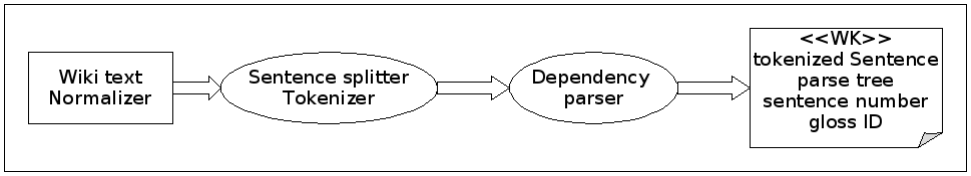

Figure 4.4: The preprocessing workflow of wiktionary gloss.

Wiktionary is written collaboratively by volunteers using wiki software. We use only glosses of nouns to learn semantic relations, because a gloss explains the meaning of a word, so that it likely indicates the relation between closely related words. The whole data is again splitted into training dataset and testing dataset with ratio 7:3. There are 86659 senses comprising 92592 sentences in the training dataset. The testing part contains 34638 senses with 20364 sentences. The preprocessing is different like blog because it uses wiki marks for editing and contains less noise. The first step is replaced by a **Wiki text normalization** block. This block removes first all wiki marks like wikify links, templates to get a clean plain text. Because most words are explained simply by another word or a short phrase like "A solid or hollow sphere", it is necessary to add the word of the entry at the front of each gloss in order to show the relations between words within one sentence. We add each word to the front of a gloss together with *is*. A phrase "A solid or hollow sphere" becomes a sentence "Ball is a solid or hollow sphere." Use *is* instead of *means* or other verbs because *is* is assigned to a special POS *VBE* by Minipar so that the relation between two nouns "ball" and "sphere" is represented as a relatively unique dependency path N:s:VBE:pred:N. It produces meaningful results with less noise. **Repeated words elimination** is not necessary any more because glosses contain hardly any repeated words. As a result, we get 112956 tokenized and parsed sentences.

4.2 Implementation of Coordinate Terms Learner

Two pass of all documents are required before runing the modified clustering algorithm. In the first pass, *idf* of each Minipar identified lemmas are collected. A lexicon E is build by allowing the terms with normalized *idf* within [0,0.9]. In the second pass, a co-occurrence matrix between the word under discussion and local context terms is generated. We allow two different context definitions. Besides the previous proposed one, a simple local context is two words to the left and two words to the right of a word under discussion, which is used in (Ravichandran et al., 2005) for web corpus. The simple context results in much smaller dimension of a co-occurrence matrix than the grammatical context. Therefore no lexicon is required. The result from the algorithm is a set of term clusters, whose size is determined by the threshold of vector similarity.

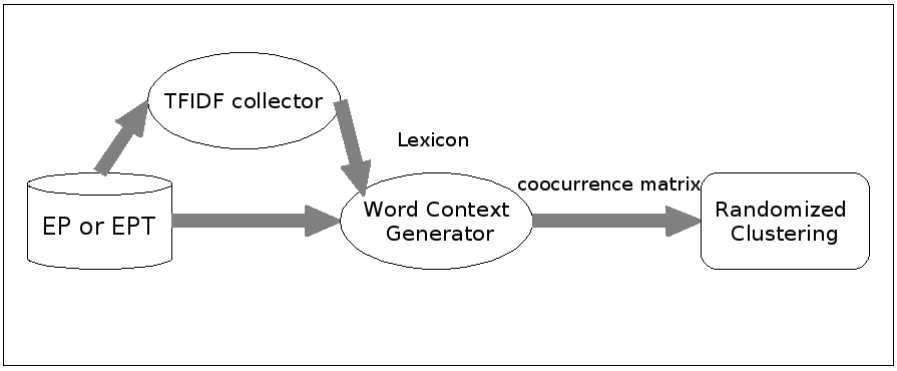

Figure 4.5: The workflow for clustering taxonomic cousins.

4.3 Implementation of Lexico-syntactic Pattern Based Classifier

4.3.1 Snow et al.

As (Snow et al., 2005), we use dependency paths to represent lexico-syntactic patterns. Prepositions are represented as nodes in a dependency graph generated by Minipar (figure 2.2). For a better representation of the semantic relation between a pair of words depending on a preposition, we apply a transformation rule to connect the prepositional complement directly to the words modified by the preposition. The new relation is labeled by the preposition. The technique is called collapsing, which is also used in stanford parser (de Marneffe et al., 2006). Figure 4.6 represents the dependency graph showed in the section 2.2.1 after collapsing.

For a better representation of conjunction, the involved dependency relations are distributed across all conjunctions to overcome the representation shortage stated in the section 2.2.1. The antecedents of a word identified by Minipar is modeled as "antecedent" relations connecting the involved words. In addition,

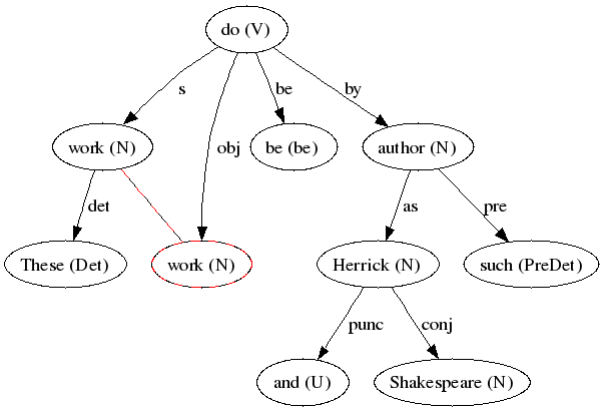

Figure 4.6: A dependency graph with collapsing, which parses a text fragment "These works are done by such authors as Herrick and Shakespeare ".

Minipar does not differentiate common nouns from pronouns. Since it makes no sense to compare a common noun with a pronoun, we add a new PRONOUN POS to the dependency graph. All shortest paths between common noun pairs are then organized into a dependency tree to save the memory consumption. Satellite links are represented as semantic neighbour nodes attached to the corresponding tail nodes.

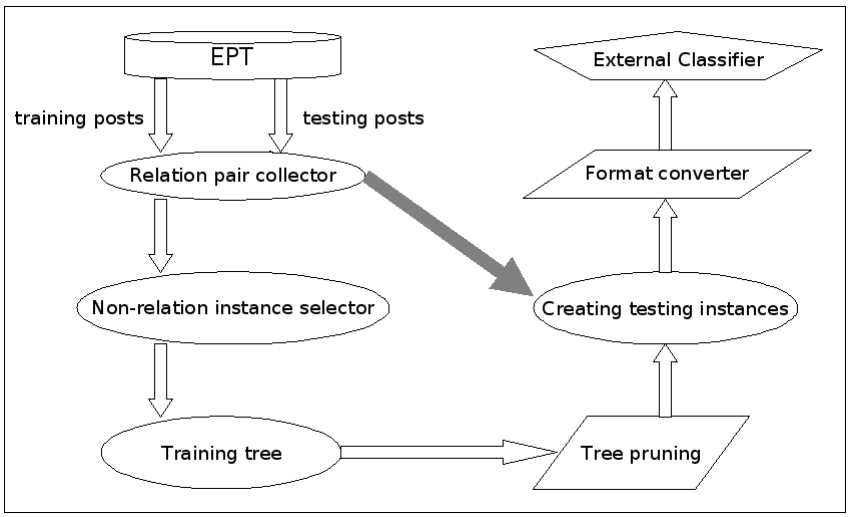

Figure 4.7: A workflow of the implementation of the method in (Snow et al., 2006)

- All unique tagged English posts are divided into a training and a testing dataset with ratio 4:1. The noun pairs with at least one noun not contained

in WordNet are considered as part of testing dataset, even if they occur in training dataset.

- Word pairs indicative relation R are collected in this step. We collect first all hypernym pairs, non-hypernym pairs and the noun pairs not contained in WordNet for each dataset. According to (Snow et al., 2005), they achieve the best performance when labeling a noun pair (n_i, n_j) as Hypernym if n_i is a n ancestor of the first sense of n_j in the WordNet taxonomy. A noun pair (n_i, n_j) is a Non-Hypernym if they are both contained in WordNet and neither noun is an ancestor of the other for any senses of either noun. The results are stored together with their occurrence in separate tables. If a noun pair is recognized as hypernym, their distance in the WordNet hierarchy is record as an additional attribute of the hypernym table.

- After the process, we maintain the ratio between hypernym and non-hypernym pairs by selecting the top 50xN most frequent non-R word pairs for training, given N word pairs of relation R.

- The first step of training is to build a dependency tree. The learner reads each tokenized sentence and the corresponding parse tree of the training dataset from database and builds a tree incrementally. It involves additional techniques for saving memory, which will be covered in section 4.5. Because some patterns like "[U_and,punc]N:conj:N[A_other,mod]" requires the encoding of co-occurrence of 2 SNs, the count of SN nodes increments only when 2 nodes take place together. Hypernym extraction requires additional steps. According to (Snow et al., 2006), they build separate classifiers for each H_{ij}^d and $d \in \{1, 2, 3, 4, 5\}$., where H_{ij}^d denotes that a sense j is d-th ancestor of sense i.

- A node selection step is applied to remove all nodes occurred less than 5 times because these words do not produce stable patterns.

- Each instance in its instance base is then converted into a real valued sparse vector. For a tail node only subtree, which is mostly the cases, the access count C_A is the value of vector element. In the conversion, each semantic neighbour is treated as a single attribute in order to differentiate the pattern "such N as N" from "N as N". If a tail node contains such a SN, the count of SN is subtracted from its own count when it serves as a feature. It works because there is at most one SN node of a subtree and two co-occurred SN nodes are viewed as one node.

- Test data builds another instance base of the trained dependency tree. The value of a feature is a matched tree path from root to tail node of a trained dependency tree. In testing data building, each matching dependency paths from testing documents is stored in a new instance base. And it is subsequently converted to a sparse matrix the same as in the training phase.

- (Snow et al., 2006) shows that logistic regression classifier performs best on the data discretized by exponentially increasing buckets {(0,1],(1,2],(2,4],...}. All current frequency count based features are therefore transformed into

a sequence of binary features. The testing matrix contains a lot of noun pairs, which are not contained in WordNet. In this step, these instances are simply relabeled as Non-Hypernyms. The sparse logistic regression classier from Auton lab (Komarek and Moore, 2003) is one of the fastest classifiers for large sparse data. It can do this classification task within 1 minute, whereas other ones like the corresponding one in yale (RapidMiner) (Mierswa et al., 2006) spend days on the same task. The format converter supports also the transformation into the yale format and the sparse format used by SVM^{light} (Joachims, 1999)

4.3.2 Bayesian Network Estimation

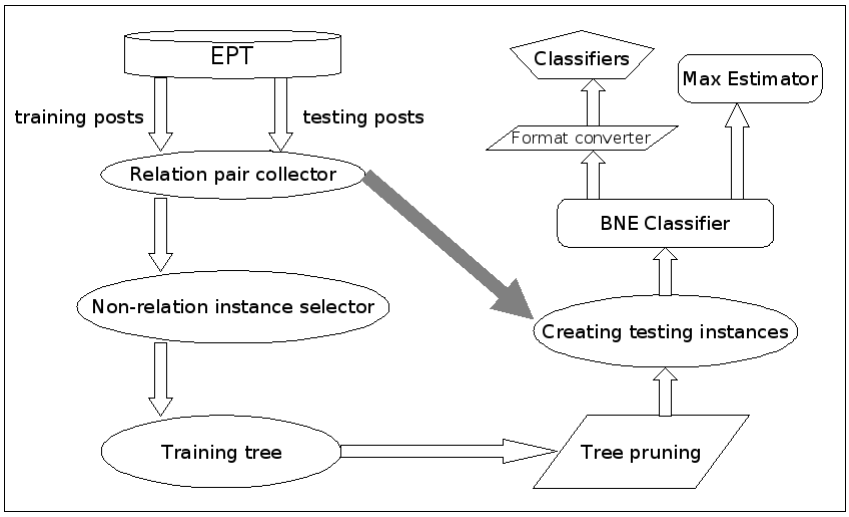

Figure 4.8: The workflow of Bayesian Network Estimation introduced in section 3.2.2

BNE differs from the previous approach in the way of feature representation and classification because it allows any types of SN nodes. The prediction task is accomplished by our Bayesian Network classifier presented previously. Another difference lies in tree pruning, which prunes all tree nodes occur less than a minimal frequency. After classification, each subtree vector of testing instances is converted into a real valued vector, whose value is the probability of a dependency path indicative a relation R. One choice is to use max estimator choosing the max value of a vector as the final probability of a instance. As the sparse feature matrix can be converted to data formats of Autonlab classifiers and SVM^{light}, another choice is to employ their classifiers to learn the "confidence" vector after BNE estimation.

4.4 Implementation of Topic Related Relation Learner

4.4.1 Tag Ontology Extraction

In the task of tag clustering based on context, each tag is stemmed by tree tagger and the corresponding context terms are stemmed by Snowball. The terms with extreme high IDF values (higher than 0.9) from a blog post are removed from tag vectors because they are mostly single-use words and their existence can easily lead to memory problem of randomized clustering.

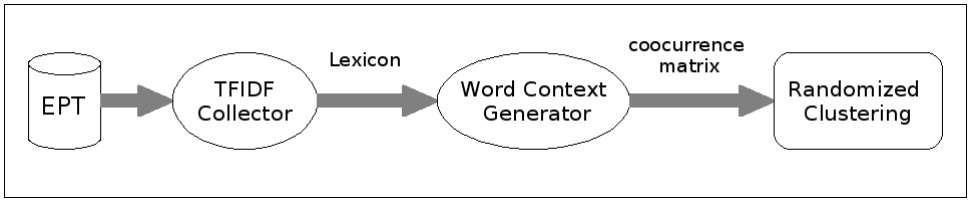

Figure 4.9: The workflow of blog tag clustering

We use ADtree (Moore and Lee, 1998) to facilitate the discovery of association rules. It is a caching data structure for doing counting in constant time. Given R records and m attributes $A = \{A_1, A_2, ..., A_m\}$, a query is a set of (*attribute* = *value*) pairs. Each ADnode (shown as rectangle in figure 4.10) represents a query and stores the number of records that match the query. A "Vary nodes" D of attribute A_i (shown as ovals), which is a child of an ADnode, has again one child for each of the n_i values of A_i, if the arity of A_i is n_i. The kth child ADnode of D represents the same query as its Vary node's parent with additional constraint that $A_i = k$. In order to reduce redundancy, if a ADnode has Vary node of attribute A_i as its parent, it can only have Vary node children representing attributes with higher index. A lookup to get the number of matched records of a query $(A_2 = 3, A_4 = 1)$ can follow the path in the tree: $VaryA_2 \rightarrow A_2 = 3 \rightarrow VaryA_4 \rightarrow A_4 = 1$. The count is obtained from the resulting node. Notice that such a tree is quite huge, for M binary features 2^M nodes are required. So they store NULL both at the position of ADnodes that match zero records and the ones representing the most common of the values of an attribute (they call it MCV).

Because each tag is modeled as a binary feature and only positive association is of our interest, a complete ADtree is not necessary. We assume that absence of a tag is MCV, so only the co-occurrence information is need to be stored into the tree. Another simplification is that we remove vary nodes from the tree because they have just one Not-NULL child. Therefore, the building of tree can be accomplished within two scans of database. Let a record be a set of attributes $A_i, A_{i+1}..., A_k$ having positive value in increasing order of index and a_i denote an ADnode of attribute A_i, the modified algorithm is given here:

```
buildADtree()
```

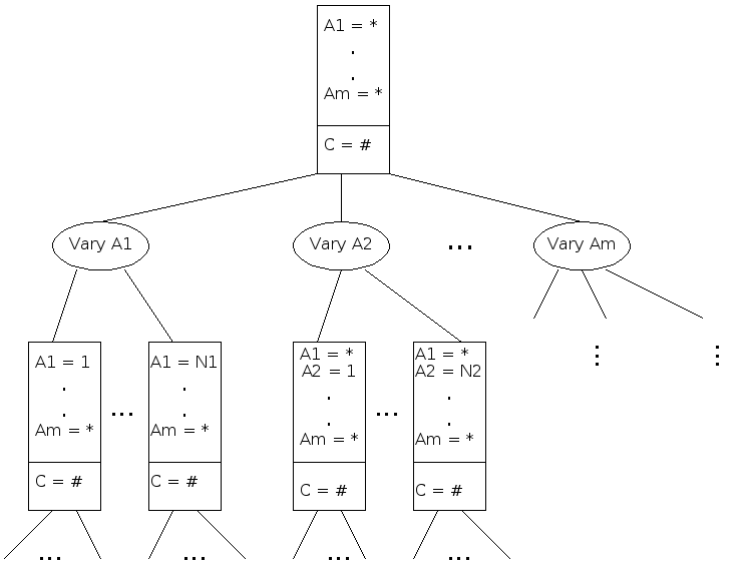

Figure 4.10: The top ADnodes of an ADtree. Each vary node represents a contingency table.

```
termSet = collect terms occurring at least m times.
for each record r in database:
   let c = the intersection of present terms of r and termSet
   keep c sorted in increasing order
   incrementTreeCount(c)
```

The algorithm begins with a scan through database to calculate the collection frequency of terms. This is the first step of Apriori algorithm (Agrawal and Srikant, 1994) that does not take terms without minimal *support* into account. It saves large amount of memory because there are large portion of single-use terms. The function incrementTreeCount is defined as:

```
incrementTreeCount(c)
   while c is not empty:
      append tree root to queue Q
      sortedSetA = c
      while sortedSetA is not empty:
        currentNode = get first element of Q
        increment count of currentNode
        remove first element of sortedSetA
        for element e in sortedSetA:
           K = get or create child of currentNode matching e
           Add K to the end of Q
      remove first element from c
```

The function goes through all combinations of input set and increments the count correspondingly. This step takes $O(2^a)$ time where a is the number of

positive values of a record. Because a database record has about 4.2 tags on average, so that the computational complexity is in fact closed to $(16N)$ where N is the number of database records.

After building the ADtree, all frequent itemsets are found by depth first traverse throughout the tree. It worth noticing that not every node need to be accessed. Since the count decreases as the tree depth increases, search stops at nodes with count lower than specified minimal *support*. In each frequent itemset, $equality^{\epsilon}$ and *conviction* are verified subsequently. *conviction* will be checked only if $equality^{\epsilon}$ is lower than the given threshold. As *conviction* defines directional rules, the brute-force method that finds all rules within a frequent itemset requires exponential computation time. A simple method suggested in (Witten and Frank, 2005) is to find first single-consequent rules because double-consequent rules hold only if the single consequent rules are true. If rules with more consequents are found only from rules with less consequents, the required computation time is enormously reduced.

4.4.2 Keyphrase Extraction

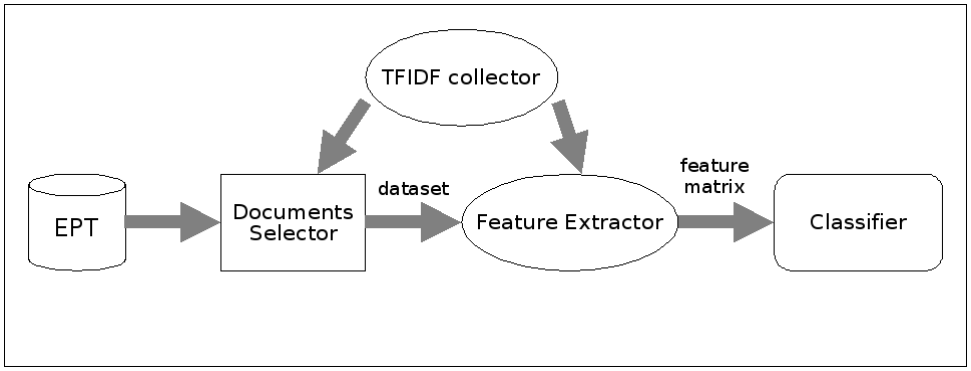

Figure 4.11: The workflow diagram of keywords extraction.

As mentioned in the last chapter, lots of annotated posts do not contain tags as part of their main contents since bloggers intend to use words of general categories to annotate web content. The existence of ambiguous tags like "toread", "fun" should not be included as target class because only well edited keywords are of our interest. Document selector fulfills the two requirements in two steps. Ambiguous tags are "ambiguous" because they are used to annotate posts of various topics. Just like measuring the quality of a cluster we use the average squared distance to filter out the posts annotating dissimilar posts. Let D be all associated posts of a tag t and n be the size of D, the criterion is defined as:

$$avgdist(t) = \frac{1}{n^2} \sum_{a \in D} \sum_{b \in D} cosine(a, b) \tag{4.1}$$

where *cosine* measures the cosine distance between two normalized TFxIDF document vectors. Only tags whose associated posts have *avgdist* over a given

threshold are regarded as candidate tags. The tags are removed from each annotated post in advance. Then tags and posts are stemmed by Snowball before checking if a tag exists in its referred post. A post is selected only if it contains minimal number of tags which are candidate tags and reoccurred in the text. A tag fulfilling the minimal condition is referred as "index terms" in the latter part.

Since the tokens and POS information already exist in the database, we use directly POS category identified by Minipar to collect candidate terms. N-grams of patterns "N", "A N", "N N","A U N"[1], "N U N","A Prep N" and "N Prep N" comprise the candidate term set. It covers more that 80% percent of the selected tags. The major loss is resulted from the single word used in sentences longer than 60 words. Most features are trivial to get, only the ontology feature is not obtained from clustering results and association rule sets but directly from the transformed PMI co-occurrence matrix and ADtree, because the temp results allow more flexibility and *interest* measure is not contained in the rule set.

4.5 Performance Issues

Processing large-scale dataset in restricted time is one of the biggest challenges in this thesis because weblog corpus consists of over 50 GB data. With a standard PC having maximal 2 GB memory a good balance between space and time should be achieved. Even the simplest task, storage of Hypernym and Non-Hypernym pairs became a problem at the very beginning. B-tree as the default indexing method of Mysql database showed quite a low performance when table entries grew up to 1 million. Hash indexing solves the problem although it worked a little slower than B-Tree at beginning phase.

"Out of memory" is one of the most frequently emerging problems during development. Carefully designed data structure and algorithmic improvement are the keys to the solution. Sparse vector and sparse matrix structure save enormous space. Dependency tree consumes much less memory by storing identical nodes of a dependency path only once. Using minimal data type as attributes can lead to further optimization. E.g. use byte instead of unicode string to represent the POS of a path node consumes at most 1/8 of the original memory. It is especially beneficial if there are over millions of tree nodes. But there were also situations, where even the most efficient data structure can not fit into memory. The instance base of a dependency tree for hypernyms requires about 1.5 GB space without consideration of the tree itself. A *tf* matrix needs even 16 GB space for all English posts even when sparse matrix structure is used. The only solution is to embed a cache into the data structure, which saves the least used data into disk and keeps only the frequent used ones. Caching strategy is simple for write only situations like storing *tf* vectors, it simply writes all data from cache into disk when the cache is full. If read is the main action of a task, least recently used (LRU) is one of the basic strategy. Noticing that many data mining algorithms work sequentially, if indexes of data objects are assigned in the same order, a set of objects with subsequent indexes of the current object

[1] U is included only if it is not a punctuation.

can be preloaded into memory while reading them again in the same order.

Large scale NLP processing is also a time consuming task. E.g. it took over a week to filter out all non-english posts and about ten days for parsing all the English posts with Minipar, even when Minipar is one of the fastest dependency parser to our knowledge. 6 or 8 hours are mostly the least running time for most tasks like collecting hypernyms. To prevent rerunning some long tasks, the experiment system is designed to store any important temp results to let fast prototyping possible. Lots of results do not result in chaos of data management because the relatedness between various results and the original data are kept by foreign keys, which benefits a lot from the representation power and scalability of the relational model. It is also important to employ profiling tools like JProfiler and Netbean profiler to find the performance bottleneck.

Chapter 5

Experiments and Analysis

In (Brank et al., 2005), most ontology evaluation approaches are classified into four categories:

- An ontology under discussion is compared with a "golden standard", which may also be an ontology like Cyc and WordNet. Precision and recall are widely used to measure the correspondence between them. However, if the ontology is from a specific domain, preparing the golden standard requires a lot of human work. A human edited ontology always suffers from low coverage, which can only be used to evaluate part of an ontology.
- Typically, an ontology is used in certain applications or tasks to see if it improves their performance. If the outputs of a application in question are straightforward and there is a well-understood evaluation of the application, an ontology can be simply plugged into the application and the evaluation is based on the results of it. But the observation is indirect and influenced by a lot of factors because the ontology is only part or small part of an application.
- An ontology can also be compared with a source of data about a domain the ontology refers to. E.g. (Brewster et al., 2004) compares a set of ontologies against a corpus of a specific domain. They conduct a probabilistic approach to measure the overlapping of domain specific terms between an ontology and the corpus.
- The most basic approach is to evaluate an ontology by humans who try to assess how well the ontology meets the predefined criteria.

Section 5.1 compares learned coordinate terms against WordNet. Section 5.2 evaluates extracted relations against not only WordNet but also Research Cyc[1] to improve the coverage. Keywords extraction is evaluated with a hold-out testing set in section 5.3. In this section an experiment is also carried out to compare the quality of extracted keywords before and after involving tag ontology, which can be regarded as the second approach.

[1] http://research.cyc.com

5.1 Evaluation of Coordinate Terms

The algorithm used to find coordinate terms is the improved clustering method proposed in section 3.1. We use the similar method in (Pantel and Lin, 2002) that follows a "golden standard" based approach. Because weblog corpus contains a lot of words that are not covered by WordNet, we evaluate only the clusters in which at least half of the words are contained in WordNet. The words not contained in WordNet are not considered as members of a given cluster.

The evaluation method measures the percentage of output clusters corresponding to their synsets. If sim_c is over a threshold θ, it is recognized as a correct sense.

$$max\ sim_c(s, c) \geq \theta \tag{5.1}$$

Let C denote all words in a cluster c and $\#C$ be the number of words of c, sim_c between a cluster c and a synset s is defined as:

$$sim_c(s, c) = \frac{\sum_{w_i \in C} sim_w(s, w_i)}{\#C} \tag{5.2}$$

The similarity between a word w and a synset s is the maximum similarity between a synset of w and s.

$$sim_w(s, w) = max_{u \in S(w)} sim_{sense}(s, u) \tag{5.3}$$

where $S(w)$ are all synsets of a word w. The similarity between two synsets s_1 and s_2 is defined in Lin and Pantel (2002) as:

$$sim_{sense}(s_1, s_2) = \frac{2 \times logP(s)}{logP(s_1) + logP(s_2)} \tag{5.4}$$

where s is the least common subsumer which is the most specific synset that subsumes s_1 and s_2. The probability P of each synset is derived from semantic concordance files[2] provided together with WordNet 2.0, which records the frequency count of the synsets tagged in the corresponding concordances like SemCor[3]. To reflect the taxonomic hierarchy, the frequency count of a synset is propagated to all its ancestors.

As the precision of a word w is the percentage of correct clusters being assigned to, the overall precision of a clustering algorithm is the average precision of all words. From 15,000 noun clusters we obtain 31.42% with threshold 0.1, which is only about the half of that in (Pantel and Lin, 2002). One reason lies in relative smaller dataset (about 50,000 posts), compared to 1GB newspaper text used by Lin and Pantel. The most important reason lies in the nature of Web 2.0 that grassroots have inconsistent style of writting. There are a lot of compound words that contains numbers, symbols. Minipar parser, which is trained on SUSANNE corpus, has difficulty in identifying POS and lemmas properly. For instance, in the following example, "ubuntu center current" includes "current" as part of a noun. And inproper use of expressions group dissimilar words into clusters, which they do not belong to.

[2]The attribute tag_cnt of index.sense file represents the number of times a sense is tagged in various semantic concordance files.

[3]http://multisemcor.itc.it/semcor.php

debian ubuntu , linux terminal server project
ubuntu center current , tech ubuntu center alpha 1
gnome gui , flavors - ubuntu

Table 5.1: Three noun clusters containing the word "ubuntu". Each row contains a cluster and cluster members are separated by comma.

5.2 Experiments of Lexico-syntactic Pattern Based Learning

As the last section evaluates the 1-ary relations, we evaluate here binary relation instances extracted by the baseline method and BNE against golden standards. As discussed in section 4.3, all tagged posts are splitted into training and testing datasets. WordNet is used to label word pairs in the datasets with specific relations which are both contained in WordNet. In the evaluation of extracted hypernyms, Research Cyc, which is the world's largest and most complete general knowledge base, is used to improve the coverage of the evaluation. Cyc contains a huge amount of concepts and facts. A semantic relation between two concepts is expressed through a logical assertion like "Human #isa Animal", where #isa is a predicate asserting the subsumption relation between two concepts "human" and "animal".

5.2.1 Hypernym Extraction

We apply first the baseline method on testing sets of weblog corpus and Wiktionary glosses separately. The precision and recall is evaluated by the WordNet labeled word pairs. Poor performance is achieved on both datasets. The logistic regression classifier from Autonlab classifies nearly every relations as Non-Hypernyms. Nearly every noun pairs have over 99% confidence as Non-Hypernyms. After analysis of small amount data, five possibilities are found as reasons of the misclassification.

1. A possible positive pattern is represented as a dependency path with high probability indicative a relation. Hearst's patterns are examples of such patterns. Low recall indicates that such patterns may be seldomly used to describe a relation. If the positive patterns existing in our datasets are too sparse, the used classifiers may fail to classify positive instances correctly because they assume false negative and false positive have equal cost and they are designed to minimize overall misclassification cost. This can be explored by ranking each *bridge* in the training dataset, which is equivalent to rank all tail nodes of the trained dependency tree according to $p(n_a)$ calculated by equation 3.1. It is also a way of pattern discovery. The ranking list of H_{ij}^2 of wiktionary gloss shows that no pattern occurs more than 5 times with $p(n_a)$ over 0.5. The best frequent postive pattern "N:s:VBE:pred:N:as opposed to:N" occurs only 14 times and has a score of 0.2857. Although there are 523 patterns containing the keyword "such as", only five of them have $p(n_a)$ over 0.2 but all such patterns occur no more than five times.

2. Another possible Hypernym pattern is a set of co-occurred dependency paths representing a positive example. They can build such a pattern even if none of them indicates alone the relation in question with high confidence. If the number of co-occurred paths are quite small for positive examples, it will be difficult to draw a decision boundary. After inspecting the blog testing data of H_{ij}^2, it is found that the average number of features in is only 1.8 with standard deviation 1.83, whereas Wiktionary has even a lower average number of features of 1.27, if only first senses are chosen to label positive examples.

3. The reason to use WordNet is to label each lexico-syntactic pattern indicative a relation in question. If two words not recognized as the specific relation are used in a positive pattern frequently, the pattern can not be identified. The grassroots writers of web 2.0 can misuse the expressions or they could fail to recognize the relation between two words as precisely as professionals. (Snow et al., 2007) argues that the relations in WordNet are too fine defined, which can also lead to poor performance. We find clues from the lexico-syntactic pattern "N:s:VBE:pred:N" we built in Wiktionary corpus by attaching the word of entry in front of its gloss. The $p(n_a)$ of the pattern is only 0.013 although it takes place 83879 times. It is far from intuitive estimation. After analysis of some sample pages from en.wiktionary.com, we find out that the average correspondence between WordNet and Wiktionary is low but still much higher than 0.013. For example, the word "catch" finds only two Hypernyms covered by WordNet at the first occurrence of a common noun among its all seven meanings. But "catch" is not a "problem", "clasp" or "find" in WordNet, although the Wiktionary writers think they are. So precision labeled by WordNet would be 40%, it generates three false negative examples. And it is noticed that none of the correspondence comes from the first sense of WordNet. Another type of mismatching is that the sibling in WordNet can be written in Wiktionary as Hypernym relations. In wiktionary, a cat can be "an enthusiast or player of jazz" but enthusiast is not an ancestor of cat at any level. However, they share a least common subsumer "person" in WordNet.

4. The lexico-syntactic pattern representations in (Snow et al., 2006) may provide insufficient information for small amount data. Incorporating context information is a way towards further improvement.

5. Several posts in weblog corpus contain some advertisements or commercial information which are automatic attached by web agenten. Such kind of sentences are nearly identical. The failed recognition of compound words in such sentences can therefore lead to repeated false examples in the training and testing phases. For instance, "power" is a direct ancestor of "control" in WordNet, but they are not of Hypernym relation in the phrase "Age of Empires III-Real-Time Strategy Game Control a European power on a quest to colonize and conquer the New World.". We partly solve the problem by means of Minipar lemmatization and stanford named entity recognizer. In the above example, Minipar identifies "Empires III-Real-Time Strategy Game Control" as the lemma of "control".

(Suchanek et al., 2006a) gives a theoretical upper bound of classification errors in terms of the *quality* and *allotment* of a pattern p. They show that a good choice of positive examples and counterexamples decreases the probability of misclassification. If a pattern indicates strongly a relation, it is also unlikely to have a wrong label. As the number of instances increases, the error bound converges to zero, so the dataset should be large enough to get a reasonable result.

We establish a set of experiments to find the optimal choice of examples. Among them we obtain the best results on Wiktonary corpus when we consider all senses of a word and take H_{ij}^d, $d \in [1,5]$ as positive examples. For weblog corpus, only H_{ij}^d, $d \in [1,3]$ are regarded as Hypernym. Two choices of counterexamples are employed in the experiments. One is the same as in (Snow et al., 2005) that any relations not belonging to H_{ij}^d, $d \in [0,\infty)$. The other does not take certain taxonomic cousins C^dij into account, where C^dij denotes a pair of taxonomic counsins with maximal distance d to their least common subsumer. Naive Bayes of Autonlab is also used in the baseline method in order to isolate the dependency on logistic regression classifier. In addition, Bayesian Network Estimation introduced in the chapter 3 is applied to explore the usefulness of incorporating context information.

The evaluation follows a coarse-grained and a fine-grained approaches. In coarse-grained evaluation, the taxonomic cousins C_{ij}^1 and C_{ij}^2 in testing datasets are not labeled as counterexamples, whereas all taxonomic cousins are Non-Hypernyms in the fine-grained evaluation. To improve the coverage, Research Cyc is used to cover the estimated noun pairs that are not contained in WordNet. The nouns in form of word constants are first mapped into Cyc concepts by means of predicates #$denotation, #$multiWordString and #$compoundString. The last two predicates associate compound words with concept constants. Hypernymy in Cyc is addressed by predicates #$isa and #$genls, which are equivalent to instance-of and subsumption relations. In addition, different thresholdings θ of prediction confidence are set to allow different selection of estimated postive examples. According to the proposed statistical model in (Snow et al., 2006), k of equation 2.12 is a factor adjusting the proportion of estimated relations added into a taxonomy T. This is equivalent to allow different thresholding over predicted confidence. So results are also given separately when $P(R_{ij} \in T \mid E_{ij}^R)$ is larger than 0.1, 0.2 and 0.5. As usual, precision, recall, F-measure and accuracy are defined as follows:

$$\begin{array}{ll} pecision = \frac{TP}{TP+FP} & recall = \frac{TP}{TP+FN} \\ accuracy = \frac{TP+TN}{TP+TN+FP+FN} & F-measure = \frac{2\times recall\times precision}{recall+precision} \end{array}$$

where TP, FP, TN, TN stand for *true positive*, *false positive*, *true negative* and *true negative*.

The promising results of evaluation are given in appendix A. Among the different options of the baseline method, the one using logistic regression classifier and including all taxonomic cousins as counterexamples shows the best performance in both corpora. Naive Bayes achieves a better recall without C_{ij}^1 and C_{ij}^2. BNE performs best also without C_{ij}^1 and C_{ij}^2. This demonstrates the simi-

word A	word B	confidence
company	organization	21.65%
form	knowledge	21.16%
device	phone	20.46%
cancer	disease	23.64%
jesus	god	25.71%
china	country	39.52%
bolivia	country	20.44%

Table 5.2: The top seven hypernyms found by the baseline method in the weblog corpus.

larity between the proposed BNE and Naive Bayes, which is in fact an improved Naive Bayes considering more dependencies. The reduction of noisy examples help them to get better estimation.

Incorporating context information does not show strong advantage over the baseline method in terms of precision and recall evaluated by "golden standard". The reason lies mainly in the coverage of WordNet and Research Cyc in case of Wiktionary corpus. At threshold 0.2, which has a balanced precision and recall, BNE finds 34,011 hypernym candidates, whereas the baseline method identifies only 6,977 potential hypernyms. At the same level, the F-Measure of BNE (42.83%) is also slightly higher than the baseline method (42.12%) in the coarse-grained evaluation. However, the baseline method show its strength in weblog corpus, where it finds the following seven hypernyms presented in table 5.2.1 with 100% precision, if the confidence cut is set to 0.2. BNE can reach the similar performance in terms of F-Measure when logistic regression classifier is employed instead of the max Estimator to make prediction on confidence space. The F-Measure of BNE is about 00.67%, which is higher than 00.22% of the baseline method. At threshold 0.1, the F-Measure of BNE is even four times higher than that of the baseline method at cost of 4% reduction of precision. Incorporating context information can no doubt capture more lexico-syntactic patterns and results in higher recall. It introduces also more noise and dependent features into feature space. BNE has still too strong independence assumption and therefore can't achieve quite high precision. The large difference of precision with the baseline method between logistic regression classifier and naive bayes classifier on weblog corpus shows also the weakness of the simple Bayesian classifiers.

Max Estimator performs well if the density of positive patterns is high and data contains little noise. It can at best indicate if there is a pattern highly indicative the relation in question. Previous experiments show that max estimator performs equally well as logistic regression classifier on Wiktionary corpus. However, the assumption of max estimator that "the semantic relation is most indicated by the strongest expression used to describe it." does not hold any more if the data become more noisy. Compared to that the co-occurrence of multiple lexico-syntactic patterns builds more stable patterns of semantic relations from a large corpus.

The different evaluation results on different corpora reveals the characteristics of data. Methods worked on Wiktionary has always much higher recall than weblog, since the online dictionary contains 11,243 hypernyms in only 112,956 sentences, whereas weblog corpus has only 4576 hypernyms among all 418,610 sentences. And Wiktionary is also rich in expressions that explains the word relations, which can be considered as the density of positive patterns. Therefore, BNE with Max estimator performs well on that corpus, since it is good at extracting relations with high density and little noise. Besides relation and pattern density, they varies also in frequency of reoccurrence. Since the high reoccurrence in weblog corpus allows the large size of co-occurred dependency paths, the baseline method achieves higher precision than that on Wiktionary corpus. It is the also reason why over 200 GB data is used in (Snow et al., 2006) to achieve the best result. However, there is a still a distance between our results and the results in (Snow et al., 2006). Besides the noisy nature of Web 2.0 and small data size, we can not apply taxonomic induction is another reason because the number of extracted relations from weblog corpus is too small and extending WordNet is not of our interest.

5.2.2 Meronym Extraction

In order to show the strength of lexico-syntactic pattern based classifier, we apply BNE and the baseline method to extract meronyms from Wiktionary and weblog corpus. In the training phase, a relation R_{ij} is regarded as a positive example if the first 3 synsets and their direct ancestors of i in WordNet are holonymy of any synset of j. Any relation is considered as a Non-Meronym only if any synset and their ancestors do not have meronym relationships to each other. In the evaluation phase, a pair of words are meronym if any sense or their ancestors are Meronym, otherwise they are Non-Meronym. BNE with logistic regression classifier shows overall better performance than the baseline method with logistic regression classifier. Context information plays a more important role than in Hypernym acquisition. Here it should be pointed out that the corresponding precision and recall is lower than the truth because of the low coverage of meronym relations in WordNet. For example, BNE predicts 25 examples positve[4] with confidence over 0.5 from Wiktionary. 17 of them contains city names that are not contained in WordNet. All of them are correct part of whole relations between a city and a country. It is also difficult to find such relations in Research Cyc because of no proper predicate is defined.

5.3 Experiments of Keyphrase Extraction

The first experiment is carried out to compare the our keyphrase extraction method with KEA, which is available under www.nzdl.org/Kea. We select 621 documents from 2,500 annotated English posts written on May 01 2006 using the method introduced in section 4.4.2 and split them with ratio 2.3:1. Each post is selected if it is annotated with at least three “index terms” to prevent KEA failing to find at least one “index term” due to stemming error. As a consequence, the training and testing datasets contain separately 434 and 187 documents. A tag ontology is derived from the 2500 documents excluding the

[4]All these examples are given in appendix C together with confidences

evaluation	threshold	precision	recall	accuracy
BNE (LR)	0.5	0.00%	0.00%	89.27%
	0.2	32.82%	0.67%	89.20%
	0.1	31.82%	3.18%	88.89%
Baseline (LR)	0.5	0%	0%	89.35%
	0.2	30.77%	0.54%	89.28%
	0.1	30.04%	2.66%	88.98%

Table 5.3: Evaluation results of meronyms extracted from Wiktionary corpus. (LR) means using logistic regression classifier in the corresponding method.

evaluation	threshold	precision	recall	accuracy
BNE (LR)	0.5	51.27%	0.82%	94.34%
	0.2	50.78%	0.83%	94.34%
	0.1	50.15%	0.84%	94.33%
Baseline (LR)	0.5	50%	0.00005%	94.34%
	0.2	50.94%	0.82%	94.34%
	0.1	50.15%	0.84%	94.33%

Table 5.4: Evaluation results of meronyms extracted from weblog corpus. (LR) means using logistic regression classifier in the corresponding method.

ones in testing datasets.

The evaluation module of KEA provides only mean and standard deviation of hit, which is defined as:

$$mean = \frac{\sum_{d \in D} TP}{\#D}, \quad std = \sqrt{\sum_{d \in D} (TP - mean)^2}$$

where D is a collection of testing documents and TP is the number of assigned keywords matching "index terms". In order to have a fair comparison, each document is assigned 5 keyphrases by both methods.

To show the strength of important features, our method is first evaluated with only TFxIDF, *first occurrence* and *word length* features. On that basis, another experiment is carried out with *Tag Ontology* feature to see how far a domain specific ontology can improve the performance. *Named Entity class* feature is also evaluated together with previous features. The results are given in table 5.3.

From the results we can see that POS pattern based candidate terms identification outperforms the n-grams approach of KEA. The largest improvement is achieved by incorporating tag ontology, which interprets the semantic relatedness between candidate keyphrases. It can show also the effectiveness of the learned ontology and can be considered as an indirect evaluation of tag ontology. Named Entity class leads to further improvement because bloggers intend to use person names or organization names to annotate web content. The best result in terms of F-measure is even better than that of KEA++(Medelyan and Witten, 2006) (25.2%) and comparable to the highest F-measure reported in (Hulth, 2003) (33.9%) despite of the noisy nature of folksonomy and weblog.

	recall	precision	f-measure	avg hit	std hit
KEA	n.a.	n.a.	n.a.	1.09	0.99
Basic	28.40%	26.72%	27.53%	1.36	1.11
Tag Onto	30.97%	28.96%	29.93%	1.48	1.13
NE	33.22%	31.09%	32.12%	1.59	1.10
All Features	33.67%	31.85%	32.73%	1.61	1.13

Table 5.5: Evaluation results of Keyphrase Extraction. Basic means our method is evaluated using only TFxIDF, *first occurrence* and *word length* features. Based on that, Tag Onto adds *Tag Ontology* feature. NE includes further *Named Entity class* feature. The best result is achieved with all Features.

The extracted keywords are used to extend the tag ontology extracted from the 2500 weblog posts. It is found out that the extended ontology doubles the number of relations. Interesting rules like "[wayne rooney] *Rightarrow* [world cup]" appears, which previous does not exist.

Chapter 6

Conclusion and Outlook

We consider information search as two dependent tasks, ontology learning and ontology based information search, where learning a domain-dependent ontology from Web 2.0 is the focus of this thesis. We have presented several machine learning algorithms to learn relation instances from weblog and Wiktionary in terms of distributional similarity, lexico-syntactic patterns and folksonomy. As a consequence, the learned ontology is a model (U, Σ, F) that represents a weighted semantic network connecting related terms and term sets.

The data structure *dependency tree* shows a new way to represent lexico-syntactic patterns memory and computationally efficient. By means of a simple Bayesian Network Estimation and incorporating local context information, higher f-measure is achieved compared to the state-of-art method proposed in (Snow et al., 2006). Noticing that BNE does not differ much from a Naive Bayes model, which has strong independence assumption. And it does not model co-occurrence of tree nodes well, so that it has large potential for further improvement. Since basically dependency tree is a Bayesian Network, in which the dependency between random variables is represented with edges connecting them, more advanced statistical models or methods can be employed to learn a optimized network structure. Some recent works can be found in (HECKERMAN, 1999; Goldenberg and Moore, 2004).

As folksonomy is one of the characteristics of Web 2.0, the intrinstic semantic relatedness between tags are explored to derive a tag ontology from weblog corpus. Compared to the other works in this area, it is the first work to provide solutions to address synonymy, structured browsing and low coverage of tags all in the same framework. Interpreting tag structure learning as a task of association rule discovery allows to find correlation between two different term sets not just between two terms as in (Schmitz, 2006; Heymann and Garcia-Molina). The way to find similar tags based on their context transforms document similarity to tag similarity, which is able to group more similar tags than those works (Begelman et al., 2006; Heymann and Garcia-Molina) considering each document as distinct objects. Furthermore, the tags based keyphrase extraction method expands the power of folksonomy to unannotated documents, which can also be viewed as a tag recommentation system. Since tag ontology serves as one of the feature of keyphrase extraction procedure, whose result enrichs again the

existing ontology, an iterative process like Boosting can be designed to improve ontology learning. If a seed of ontology is given, it can be applied to any other free texts.

Grassroots content creators of Web 2.0 applications determine the noisy nature of contributed information. As previous experiments shown, most of learned ontologies from this domain have lower quality than the others derived from e.g. newswire corpora if it is compared with a broad coverage gold standard. Therefore, more noise removal preprocessing procedures are required than processing newswire like corpora. It is also worth noticing that the evaluation against a golden standard can not demanstrate fully the power of learned ontology due to its low coverage because the agreement between Web 2.0 and current broad-coverage ontologies is low. It is therefore a high motivation to learn ontologies from Web 2.0 that overcome the coverage shortage of the manual edited ones. In addition, the information from Web 2.0 domain are always huge and fast-growing which demands high efficiency of involved algorithms. The work in this thesis has shown how to modify the state of art methods to employ them on a standard workstation PC.

One application of learned ontologies is topic detection. Since the topic related and semantic related terms learned from the domain builds a semantic graph, the state of art graph clustering algorithms can be used to find the tightly connected subgraphs representing same topics. (Dhillon et al., 2005) establish a mathematical connection between spectral clustering and kernel k-means, so that their algorithm find clusters without any eigenvector computation. Correlation clustering algorithm proposed in (Bansal et al., 2004) does not even need to specify the number of clusters. The relevance between topic clusters and documents can be measured by their vector distance, if a topic cluster is represented as a vector of its member nodes, in which the value of a node is the sum of the weight of its edges. Therefore, it allows multiple memberships of a document. Following the same idea, if a query is mapped into the ontology and forms a centroid distributed in a cluster or between several clusters, the relevance scores between the centroid and documents in question is determined by a distance function considering semantic related terms of query in addition to the vector distance between them. Query is not a set of keywords anymore but expanded to a semantic network. The scores can be directly used for ranking, query suggestion or as a feature of a ranking model like the ranking SVM (Joachims, 2002) to obtain a domain specific ranking function. In this way, IR systems are improved in terms of incorporating semantic knowledge.

Bibliography

R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 1215:487499, 1994.

R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2):207–216, 1993.

J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998, 1998.

N. Bansal, A. Blum, and S. Chawla. Correlation Clustering. *Machine Learning*, 56(1):89–113, 2004.

G. Begelman, P. Keller, and F. Smadja. Automated Tag Clustering: Improving search and exploration in the tag space. *Proc. of the Collaborative Web Tagging Workshop at WWW*, 6, 2006.

J. Brank, M. Grobelnik, and D. Mladenic. A survey of ontology evaluation techniques. *Proc. of the 8th International mulit-conference Information Society IS*, pages 166–169, 2005.

C. Brewster, H. Alani, S. Dasmahapatra, and Y. Wilks. Data driven ontology evaluation. *Proceedings of LREC*, 2004, 2004.

S. Brin, R. Motwani, J.D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 255–264, 1997.

P. Buitelaar, P. Cimiano, and B. Magnini. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press, 2005.

P. Castells, M. Fernandez, and D. Vallet. An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):261–272, 2007.

W.B. Cavnar and J.M. Trenkle. N-Gram-Based Text Categorization. *Ann Arbor MI*, 48113:4001.

S. Cederberg and D. Widdows. Using LSA and Noun Coordination Information to Improve the Precision and Recall of Automatic Hyponymy Extraction. *Proc. of CoNLL*, pages 111–118, 2003.

M.S. Charikar. Similarity estimation techniques from rounding algorithms. *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

A. Chowdhury, O. Frieder, D. Grossman, and M.C. McCabe. Collection statistics for fast duplicate document detection. *ACM Transactions on Information Systems (TOIS)*, 20(2):171–191, 2002.

C. Clifton, R. Cooley, and J. Rennie. TopCat: data mining for topic identification in a text corpus. *Knowledge and Data Engineering, IEEE Transactions on*, 16(8):949–964, 2004.

M.C. de Marneffe, B. MacCartney, and C.D. Manning. Generating typed dependency parses from phrase structure parses. *LREC 2006*, 2006.

I. Dhillon, Y. Guan, and B. Kulis. A fast kernel-based multilevel algorithm for graph clustering. *Conference on Knowledge Discovery in Data*, pages 629–634, 2005.

R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. In *Artificial Intelligence*, volume 165, 2005.

J.R. Finkel, T. Grenager, and C. Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Ann Arbor*, 100, 2005.

R. Girju, A. Badulescu, and D. Moldovan. Learning semantic constraints for the automatic discovery of part-whole relations. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 1–8, 2003.

M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

A. Goldenberg and A. Moore. Tractable learning of large Bayes net structures from sparse data. *ACM International Conference Proceeding Series*, 2004.

T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, 1992.

D. HECKERMAN. A TUTORIAL ON LEARNING WITH BAYESIAN NETWORKS. *Learning in Graphical Models*, 1999.

P. Heymann and H. Garcia-Molina. Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems. Technical report, Technical Report 2006-10, Computer Science Department, April 2006.

D. Hindle. Noun classification from predicate-argument structures. *Proceedings of the 28th conference on Association for Computational Linguistics*, pages 268–275, 1990.

Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, New York, NY, USA, 1999. ACM Press. ISBN 1-58113-096-1.

A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223, 2003.

A. Hulth and B.B. Megyesi. A study on automatically extracted keywords in text categorization. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 537–544, 2006.

P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

T. Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning table of contents*, pages 169–184, 1999.

T. Joachims. Evaluating retrieval performance using clickthrough data. *Proceedings of the SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, pages 12–15, 2002.

L. Khan, D. McLeod, and E. Hovy. Retrieval effectiveness of an ontology-based model for information selection. *The VLDB Journal The International Journal on Very Large Data Bases*, 13(1):71–85, 2004.

Aleksander Kolcz, Abdur Chowdhury, and Joshua Alspector. Improved robustness of signature-based near-replica detection via lexicon randomization. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 605–610, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-888-1.

P. Komarek and A. Moore. Fast Robust Logistic Regression for Large Sparse Datasets with Binary Outputs. *Artificial Intelligence and Statistics*, 2003.

D. Lin. PRINCIPAR: an efficient, broad-coverage, principle-based parser. *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 482–488, 1994.

D. Lin. Automatic retrieval and clustering of similar words. *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774, 1998a.

D. Lin. Dependency-based evaluation of MINIPAR. *Workshop on the Evaluation of Parsing Systems*, pages 317–330, 1998b.

D. Lin. An information-theoretic definition of similarity. *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304, 1998c.

D. Lin and P. Pantel. Concept discovery from text. *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7, 2002.

A. Mathes. Folksonomies-Cooperative Classification and Communication Through Shared Metadata. *Computer Mediated Communication, LIS590CMC (Doctoral Seminar), Graduate School of Library and Information Science, University of Illinois Urbana-Champaign, December*, 2004.

O. Medelyan and I.H. Witten. Thesaurus based automatic keyphrase indexing. *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 296–297, 2006.

I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. YALE: rapid prototyping for complex data mining tasks. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, 2006.

R. Mihalcea and P. Tarau. TextRank–bringing order into texts. *vertex*, 4:6.

G.A. Miller. WordNet: A Lexical Database for English. *COMMUNICATIONS OF THE ACM*, 38(11):39, 1995.

A. Moore and MS Lee. Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets. *Arxiv preprint cs.AI/9803102*, 1998.

T. OReilly. What is Web 2.0. *Design Patterns and Business Models for the Next Generation of Software*, 30:2005, 2005.

P. Pantel and D. Lin. Discovering word senses from text. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619, 2002.

B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. KIM–Semantic Annotation Platform.

M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 622–629, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

Luigi Canali De Rossi. Folksonomies: Tags strengths, weaknesses and how to make them work. 2006. URL `http://www.masternewmedia.org/news/2006/02/01/folksonomies_tags_strengths_weaknesses_and.`

G. Salton, A. Wong, and CS Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

H. Schmid. Probabilistic part-of-speech tagging using decision trees. *Proceedings of International Conference on New Methods in Language Processing*, 12, 1994.

P. Schmitz. Inducing ontology from flickr tags. *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland, May*, 2006.

Clay Shirky. Ontology is overrated: Categories, links, and tags. 2005. URL `http://shirky.com/writings/ontology_overrated.html`.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*, 2005.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *COLING/ACL*, 2006.

Rion Snow, Sushant Prakash, Daniel Jurafsky, and Andrew Y. Ng. Learning to merge word senses. In *EMNLP*, 2007.

Fabian Suchanek, Georgiana Ifrim, and Gerhard Weikum. Combining linguistic and statistical analysis to extract relations from web documents. Research Report MPI-I-2006-5-004, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, March 2006a.

Fabian M. Suchanek, Georgiana Ifrim, and Gerhard Weikum. Combining linguistic and statistical analysis to extract relations from web documents. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 712–717, New York, NY, USA, 2006b. ACM. ISBN 1-59593-339-5.

Peter D. Turney. Expressing implicit semantic relations without supervision. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 313–320, Morristown, NJ, USA, 2006a. Association for Computational Linguistics.

Peter D. Turney. Similarity of Semantic Relations. *Computational Linguistics*, 32(3):379–416, 2006b.

I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

I.H. Witten, G.W. Paynter, E. Frank, C. Gutwin, C.G. Nevill-Manning, N.Z. Hamilton, and N.J. Piscataway. KEA: Practical Automatic Keyphrase Extraction.

L. Xu, A. Krzyzak, and E. Oja. Rival penalized competitive learning for clustering analysis, RBFnet, and curve detection. *Neural Networks, IEEE Transactions on*, 4(4):636–649, 1993.

C.X. Zhai, A. Velivelli, and B. Yu. A cross-collection mixture model for comparative text mining. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 743–748, 2004.

Y. Zhao, G. Karypis, and U. Fayyad. Hierarchical Clustering Algorithms for Document Datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168, 2005.

Appendix A

	Logistic regression			Naive Bayes		
threshold	precision	recall	accuracy	precision	recall	accuracy
0.5	50.80%	0.45%	85.57%	46.75%	1.51%	85.54%
0.2	42.64%	41.61%	83.50%	34.78%	53.72%	78.78%
0.1	35.74%	51.26%	79.66%	30.81%	63.00%	74.24%

Table A.1: Coarse-grained results of the baseline method on Wiktionary corpus. $H_{ij}^k,\ k \in [1,5]$ are positive examples. All taxonomic cousins are considered as counterexamples.

	Logistic regression			Naive Bayes		
threshold	precision	recall	accuracy	precision	recall	accuracy
0.5	46.67%	0.69%	85.56%	46.86%	1.56%	85.54%
0.2	42.61%	41.64%	83.48%	34.75%	53.83%	78.75%
0.1	35.45%	51.56%	79.46%	30.78%	63.13%	74.19%

Table A.2: Coarse-grained results of the baseline method on Wiktionary corpus. $H_{ij}^k,\ k \in [1,5]$ are positive examples. Taxonomic cousins C_{ij}^1 and C_{ij}^2 are ignored in the training phase.

Evaluation	threshold	precision	recall	accuracy
WordNet only	0.5	49.24%	1.36%	85.59%
	0.2	41.74%	43.99%	83.09%
	0.1	31.93%	61.89%	75.51%
WordNet and Cyc	0.5	48.38%	1.42%	85.44%
	0.2	44.42%	44.16%	82.85%
	0.1	31.90%	62.44%	75.15%
WordNet and Cyc (hard)	0.5	40.38%	1.42%	87.46%
	0.2	36.13%	44.16%	83.30%
	0.1	25.98%	62.44%	73.16%

Table A.3: Coarse-grained and fine-grained results with Bayesian Network Estimation on Wiktionary corpus. WordNet and Cyc (hard) indicates the results of fine-grained evaluation and others are from coarse-grained evaluation. $H_{ij}^k,\ k \in [1,5]$ are positive examples. Taxonomic cousins C_{ij}^1 and C_{ij}^2 are ignored in the training phase.

evaluation	threshold	precision	recall	accuracy
WordNet only	0.5	44.44%	0.11%	85.61%
	0.2	40.35%	45.39%	82.49%
	0.1	25.64%	71.96%	65.95%
WordNet and Cyc	0.5	42.11%	0.11%	85.47%
	0.2	40.26%	45.83%	82.26%
	0.1	25.71%	72.32%	65.63%
WordNet and Cyc (hard)	0.5	38.10%	0.11%	87.55%
	0.2	35.14%	45.83%	82.74%
	0.1	21.32%	63.34%	63.34%

Table A.4: Coarse-grained and fine-grained results with Bayesian Network Estimation on Wiktionary corpus. WordNet and Cyc (hard) indicates the results of fine-grained evaluation and others are from coarse-grained evaluation. $H_{ij}^{k},\ k \in [1,5]$ are positive examples. All taxonomic cousins are included in the training phase.

evaluation	threshold	precision	recall	accuracy
WordNet (max)	0.5	100%	0.0016%	94.93%
	0.2	14.21%	0.42%	94.83%
	0.1	10.41%	2.50%	93.98%
WordNet (LR)	0.5	42.11%	0.11%	85.47%
	0.2	45.65%	0.34%	94.94%
	0.1	23.63%	3.91%	94.50%
WordNet and Cyc (hard)	0.5	33.33%	0.0015%	98.46%
	0.2	6.49%	0.42%	98.38%
	0.1	4.43%	2.59%	97.65%

Table A.5: Coarse-grained and fine-grained results with Bayesian Network Estimation on weblog corpus. WordNet and Cyc (hard) indicates the results of fine-grained evaluation and others are from coarse-grained evaluation. Taxonomic cousins C_{ij}^{1} and C_{ij}^{2} are ignored in the training phase.

	Logistic Regression			Naive Bayes		
threshold	precision	recall	accuracy	precision	recall	accuracy
0.5	0.00%	0.00%	94.92%	21.67%	1.42%	94.73%
0.2	100.00%	0.11%	94.92%	17.08%	4.86%	93.97%
0.1	27.04%	0.86%	94.84%	11.47%	18.74%	88.52%

Table A.6: Coarse-grained results of the baseline method on weblog corpus. $H_{ij}^{k},\ k \in [1,5]$ are positive examples. Taxonomic cousins C_{ij}^{1} and C_{ij}^{2} are ignored in the training phase.

Appendix B

The following table contains top 25 meronym pairs extracted from Wiktionary corpus. The confidence values are predicted by Bayesian Network.

word A	word B	confidence
set	vertices	59.75%
england	staffordshire	83.61%
england	shropshire	89.43%
england	nottinghamshire	79.43%
england	wiltshire	79.43%
switzerland	zrich	95.44%
belgium	wallonia	71.00%
belgium	west flanders	71.00%
brazil	par	77.38%
brazil	maranho	77.38%
brazil	so paulo	77.38%
india	kerala	77.38%
india	rajasthan	86.43%
india	madhya pradesh	84.84%
india	maharastra	71.00%
aragon	aragonese	67.83%
kaliningrad	russia	77.38%
kyrgyz	kyrgyzstan	67.83%
abruzzo	italy	71.00%
dorset	england	79.43%
tooele	state	50.35%
north west frontier province	pakistan	71.41%
balochistan	pakistan	71.00%
braxy	sheep	67.83%
coset	subgroup	75.38%

Table B.1: Top 25 meronyms predicted by Bayesian Netword Estimation with confidence over 0.5

Appendix C

[technology, ethic] ⇒ [blog]
[DIGITe, blu ray] ⇒ [DIGITps]
[bill] ⇒ [enzi, himmaa]
[pop culture, democrat] ⇒ [humor]
[ontario] ⇒ [news, yahoo]
[commentary, karl rove] ⇒ [news]
[religion, stam] ⇒ [woman]
[art, thought] ⇒ [religion and philosophy]
[art, diary] ⇒ [scribe]
[voip, jingle] ⇒ [session signal]
[canada, soferet] ⇒ [sofrut]
[podcast, loan] ⇒ [credit]
[oschersleben] ⇒ [audi, oneighturbo]
[alabama, utah] ⇒ [mobile]
[rant] ⇒ [mom, child]
[spanish] ⇒ [boycott, commentary]
[journal, woman] ⇒ [diary]
[field trip] ⇒ [family, homeschooling]
[commentary] ⇒ [plamegate, cia leak investigation]
[real estate] ⇒ [real estate blogging, real estate blogs]
[tnt hd] ⇒ [abc, espn hd]
[psp, live psp] ⇒ [playstation portable]
[theatre, performance] ⇒ [guangzhou]
[marketing an online business] ⇒ [internet marketing, internet marketing online]
[vacation spa] ⇒ [main, stay spa the beach club spa]
[on, crush] ⇒ [own]
[waitress] ⇒ [seafood, red lobster]
[acim, enlightenment] ⇒ [a course in miracle]
[lonnie hodge, expats guangzhou] ⇒ [china editorial]
[china blogs, personal note] ⇒ [china editorial]

Table C.1: 30 association rules sampled randomly according to uniform distribution from all 34,738 rules identified by *conviction*. The right itemset of an association rule is present if the left itemset is utilized to annotate a post.

[scrap metal price, steel]
[enzi, himmaa]
[pad, photo a day]
[group b, paraguay]
[clip art, public domain clip art]
[own, then, crush, see, might]
[woman 2f 27s rugby, woman 2f 27s sport]
[filipina, sigepa]
[DIGITmayth, ggreat ggrail hunt, holy grail, last supper]
[confirmation, sacrament]
[culture guangzhou, expats, expats guangzhou, personal note, south china]
[chris cree, creeations]
[a consuming experience, consuming experience, improbulus]
[daviddmuir, edcompblog]
[super capacitor, super_capacitor, supercapacitor]
[scribe, soferet, sofrut, technorati, torah, religion and philosophy, safrut, stam]
[benedict xvi, ecumenism]
[funny blog, menopausal, menopause, menopause symptom]
[academe, duke]
[da, vinci]
[green lifestyle, carbon footprint, ecological footprint, greenforgood]
[sound alchemy online, soundalchemyonline]
[dtm, oschersleben]
[frenchtown, frenchtown nj]
[monad, powershell]
[broadband scandal, sheep herd]
[oclc, rlg]
[shiite, sunni]
[DIGITcity sanmateo, glu mobile]
[a virgin plea, united state of america]

Table C.2: 30 association rules sampled from 199 rules identified by $equality^{\epsilon}$. The right itemset of an association rule is present if the left itemset is utilized to annotate a post.

user review,buy online,camcorder,cybershot,user rating, underwater camera,DIGITsony w,online review,DIGITw,camera enthusiast, telephoto,neat find,consumer revi ew,pic,camera review, sony cybershot,imaging resource,DIGITsony dsc w,DIGITsony dscw,shop
uefa,nelson,hedgehog,middlesbrough,strimmers,stadium,arcadium,chili
venitha,counseling,therapy,couple,raffle marina, wet market,unhappiness,singapore expat,yacht
val,judy,chels,currently,kc,dora,john
efsa,ramazzini institute,kidney,european food safety agency, aspartame,scientific study
hania,kat,kendra,olivia,dana,greg
europe top scorer,eto o,golden boot,pichichi,luca toni,villa
codependence,libra,DIGITth house,pisces,people pleasing
star and buc,DIGIThot,power @card@,clear channel
hyde park,princess diana fountain,photo friday
monetization,greasemonkey,stylish
fon,municipal_wireless,co operative
generation gap,cultural literacy,broadcasting
gratitude,acceptance,optimism
trade press,DIGITb,business medium
blogging poet,@card@ blogging poet in @card@ day,tip for bloggers
http fug blogspot com,vesak,fug
antarctica,oceania
allergy,asthma
capital punishment,death penalty
david hick,guantanamo
cafe commons,soonae and jong
web base note,posticky
scheme,lambda calculus
deco,giuly
godwins law,marx
anderson cooper,gloria vanderbilt
speed,slow
jameson,jenna
north sea,haar

Table C.3: 30 tag clusters sampled according to uniform distribution from all 3,546 clusters. Each cluster is represented as a row and cluster members are separated by comma.

Appendix D

The explanation of grammatical notations is based on the README file of Minipar. The meanings of part of speech categories referred in this thesis are explained as follows:

Det: Determiners

PreDet: Pre-determiners

C: Clauses

I: Inflectional Phrases

V: Verb and Verb Phrases

N: Noun and Noun Phrases

P: Preposition and Preposition Phrases

A: Adjective/Adverbs

Have: have

The following is a list of the referred grammatical relationships in Minipar.

appo: appositive e.g. “ACME president, P.W. Buckman”

be: is e.g. “be sleeping”

gen: genitive e.g. “Jane’s uncle”

have: have e.g. “have disappeared”

mod: the relationship between a word and its adjunct modifier

pcomp-n: nominal complement of prepositions

pred: predicate of a clause

obj: object of verbs

subj: subject of verbs

s: surface subject