NLP-BASED EXTRACTION OF ONTOLOGY INFORMATION FROM SCIENTIFIC PAPERS ON LANGUAGE TECHNOLOGY

Master's Thesis

Prepared under supervision of

Prof. Hans Uszkoreit

Dr. Ulrich Schaefer

Magdalena Wolska

Pham The Nghia

March 2011

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged. Saarbruecken, 29 March 2011

(Pham The Nghia)

Abstract

Ontology is an important resource in Computional Linguistics. Currently, all of the available ontologies are constructed manually. Ontology learning is a task of extracting ontology information from different type of resources, especially from text. Current approaches use surface patterns or dependency patterns. This thesis combines syntactic pattern with more linguistics information to extract ontology information. Additionally, we use GWAP to further improve the extracted data.

Acknowledgements

First, I would like to express my gratitude to my advisors, Ulrich Schaefer and Magdalena Wolska, for their expertise, patience and willingness to discuss problems. They have provided me with invaluable guidance.

I am also deeply grateful to Prof. Hans Uszkoreit, who is a supervisor of this thesis.

I would like to thank Dr. Valia Kordoni, Ms. Bobbye Pernice and Maria Jacob for their great support.

I also wish to thank Luthfi Darmawan, Lv Jingwei, Lilian Wanzare, Luu Phuc Loi, Tran Anh Tuan for being my friends in hours of need.

I must also acknowledge my classmates and friends for their company. Thanks to them, being a student was a great experience.

I thank COLI staff for their assistance during my master studies. Additionally, I would like to thank the EM-LCT for the financial support that I received.

Finally, I am very thankful to my family and friends for the continued emotional support which in the past two years has often proven to be the deciding factor for my success.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Ontology

Ontology, originally used in philosophy, is the study of existence. It concerns about the existence of entities, entity categorization and relations between them.

The term "'ontology"' was borrowed by Computer Scientists to describe something different. In Computer Science, ontology is a formal representation of the knowledge about a domain. It contains basic concepts in that domain and relations between them. There are some understandings of the term Ontology, which are slightly different from each other. Some people use Ontology to refer to the abstract information about classes and their relation. To some others, an ontology also contains information about instances. In this thesis, we use the term ontology in the sense that it includes also facts about instances.

The original purpose of ontology is to capture knowledge and enable certain kinds of automated reasoning. It enables people, computers and applications to share domain knowledge. An ontology provides a shared vocabulary, which can be used to model a domain. Semantic Web is one of the predominant applications to use Ontology, which allows computers to understand the content of a web-page. Nowadays, ontology is highly used in many fields of study (Medicines, Bioinformatics, Computational Physics, etc.).

1.2 Large hand-built Ontologies

Almost all of the available ontologies are built manually, they are constructed as a common source of knowledge for people working on the same domain. Some well-known ontologies include CYC, GENE Ontology and FAO Ontology. These ontologies are invaluable resources and are widely used in many practical applications.

1.2.1 Cyc

Cyc [1] stems from a large artificial intelligence project. Its main goal is to construct a comprehensive ontology and knowledge base of common sense knowledge, with the ambition to enable human-like reasoning in AI systems. OpenCyc, an open ontology, is a part of the project; it is published under an open source license. Its latest version OpenCyc 2.0, was released in July 2009. With 47,000 concepts and 306,000 facts, it is stated to be the world's largest and most complete general knowledge base and commonsense reasoning engine.

1.2.2 FAO Ontology

Geopolitical ontology [2] is a mechanism to describe, manage and exchange data related to geopolitical entities such as countries, territories, regions and other similar areas. FAO Ontology is the geopolitical ontology developed by FAO Organization; it gives basic information about contries and territories such as names, country area, land area, agricultural area, GDP, population and historical change, provides geolocation, and relationships among countries, or countries and other entities, including properties such as has border with, is predecessor of, is successor of, is administered by, has members, and is in group.

FAO ontology was built to provide the most updated geopolitical information, improve information management and facilitate standardized data sharing of geopolitical information and demonstrate the benefits of the geopolitical ontology to improve interoperability of corporate information systems

[1] http://www.cyc.com
[2] http://www.fao.org/countryprofiles/geoinfo.asp

1.2.3 Gene Ontology

Gene Ontology [3] is a collaborative effort attempting to provide a consistent descriptions of gene products in different databases, to facilitate gene product searching across databases. It started as collaboration among three model organism databases: FlyBase, the Saccharomyces Genome Database and the Mouse Genome Database. Currently many other databases, such as some world's major repositories for plant, animal and microbial genomes are included in Gene Ontology. It has become a standard tool in the bioinformatics arsenal.

1.3 Motivation

Like in many other fields, Ontology is also a desirable resource in Computational Linguistics. Ontology-based Question Answering Systems use Ontology as a main source of looking up answers. Co-reference Resolution systems can also use ontology information to improve their result. Working with a corpus on Computational Linguistics requires an ontology for Computational Linguistics itself. Even though there are many hand-built ontologies available, there is still no usable ontology on Computational Linguistics. LT-World [4], the only hand-built taxonomy for Computational Linguistics is small and lacks of a lot of information. Building an ontology for Computational Linguistics is, therefore, an important task.

Constructing an ontology is a non-trivial task. It requires exploring, gathering and formalizing knowledge. This alone is already a tedious process, since an ontology usually contains thousands or over millions of facts. The main difficulty is that an ontology needs to contain enough information to cover the domain, while it has to have a meaningful and consistent generalization. Moreover, an ontology needs to be constantly kept up-to-date. For these reasons, building an ontology is very time and cost consuming.

One way to avoid such an effort is to build a general method for automatically learning an ontology from a large set of documents in a particular domain. This

[3] http://www.geneontology.org/
[4] http://www.lt-world.org

approach is promising because these papers are written by experts and contain a lot of information, which is a valuable source of knowledge. They may express explicitly or implicitly ontology information. The motivation of this thesis is to build a method for learning automatically an ontology for Language Technology from ACL Anthology. This topic is interesting because although there are a lot of studies about ontology extraction, a satisfactory result is yet to be achieved.

1.3.1 Corpora

A corpus is a general requirement for natural language processing applications. Extracting ontology information for Computational Linguistics also requires a corpus about Computational Linguistics. The corpora used in this thesis are the ACL Anthology and WeScience.

1.3.2 ACL Anthology

The Association for Computational Linguistics (ACL) is the international scientific and professional society for people working on problems involving natural language and computation.

The ACL Anthology is a collection of papers and journals in the Computational Linguistics field. The papers are collected from conferences, such as ACL, COLING and from various workshops. The anthology consists of more than 19000 ACL papers. This vast amount of scientific text is a valuable corpus for extracting an Ontology for Computational Linguistics.

Currently, our corpus contains around 8000 ACL papers, on which OCR programs produce a satisfactory result.

1.3.3 WeScience corpus

The WeScience [5] initiative is an on-going effort to provide resources that enable eScience research and development in Computational Linguistics. WeScience has

[5] http://wiki.delph-in.net/moin/WeScience

two components: the WeScience Corpus and the WeScience Treebank. The corpus comprises a selection of Wikipedia articles in Natural Language Processing domain. These articles are pre-processed to remove irrelevant markup and then segmented into sentence-like units.

1.4 Thesis Organization

This thesis presents a NLP-based approach for ontology construction. Our approach consists of three contributions:

Chapter 2: related work

This chapter presents state-of-the-art approaches for ontology extraction as well as information extraction.

Chapter 3: Term extraction

Our proposed method for extracting terms, to build a ontology lexicon. This chapter includes extracting multi-word terms and single-word terms.

Chapter 4: Taxonomy building:

Our approach to use syntactic parser for is-a relation extraction is presented in this chapter. The method for combining linguistics and statistical information to improve the taxonomy is also discussed.

Chapter 5: Evaluation

In this chapter we present our evaluation method with Game With A Purpose (GWAP). We also talk about the experiments and discuss the performance of the system.

Chapter 6: Conclustion

We make an overall review of the thesis. We also address potential improvements and issues for further research.

Chapter 2

Background and Related work

2.1 Information Extraction

2.1.1 What is Information extraction

Information extraction (IE) is a super-field of ontology extraction. Its goal is to automatically extract structured information from unstructured documents such as web pages or emails, generally by means of natural language processing. The fast increase in the amount of information available in unstructured form makes IE more and more important. For the past decades, the World Wide Web has become a vast information source providing various type of information. However, these chunks of information are, in general, expressed in human language and cannot be collected or processed directly by a computer. If one could extract these chunks of information and integrate them into a structured form, it would form an invaluable source of information.

There are several types of information extraction: term extraction, named entity extraction, binary relation extraction, n-ary relation extraction, event extraction, answer extraction, opinion extraction and sentiment extraction.

There has been considerable work on IE. Many approaches using different levels of linguistic information have been introduced, from simple pattern based approaches using only surface form to machine learning approaches with syntactic information

such as dependency paths. Pattern-based approaches ware the first ones applied and are still the most popular in current state-of-the-art systems.

2.1.2 Pattern-based IE systems

Pattern-based IE systems usually rely on a set of patterns which express a certain kind of event or relation. For example, the following three patterns contain information about the location of an organization:

```
<ORGANIZATION> 's headquarters in <LOCATION>
<LOCATION> -based <ORGANIZATION>
<ORGANIZATION>, <LOCATION>
```

Typically, in relation extraction, before being fetched into the matching component, a document is preprocessed by a named entity recognizer (NER). A named entity found in the document will have its information attached to its surface expression. Whenever a sentence matches a pattern, the system will take out the matching text segment and extract the necessary arguments of the relation.

Linguistic level

The linguistic information used in pattern-based systems varies from shallow to deep. Early information extraction system only make use of surface patterns. In general, they use the manually constructed patterns or automatically extracted patterns to build a set of regular expressions and then match them with input sentences. Work using surface pattern includes [3] which used surface and markup patterns to extract book information (including title and author information) from a heterogeneous set of web pages.

Recent work by N. Nakashole et al.[23] also used surface patterns to extract relations. However, they do not insist on exact matching; in contrast, they use Jaccard distance to measure similarity between a pattern and a n-gram between two entities. Even though this work only use surface patterns, it turns out to get a competitive result compare to state-of-the-art IE systems.

Later work found surface patterns too restrictive but not informative enough for expressing a relation. An alternative for surface patterns is dependency paths. Figure

2.1 show an example of dependency structure of a sentence. Dependency paths have been used successfully as a replacement for surface form to build lexico-syntactic patterns for semantic relations. Dekang Lin and Patrick Pantel [19] successfully use dependency paths to discover various relations.

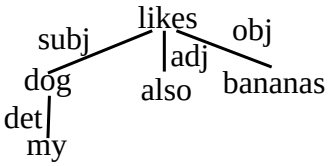

Figure 2.1: Example of dependency structure

Shallow IE methods have been proven to be sufficient for information extraction. However, for verb-centered relations which are mostly expressed by verbal expressions (e.g. "take over" relation between companies), deep grammars are more suitable to obtain exact relationships between verbs and their arguments in complex linguistic phenomena, involving, e.g., passive, free word order, long-distance dependencies and control/raising. Unfortunately, the decision of the attachment of modifiers is very difficult, therefore their analysis is often ambiguous whereas shallow analyses are straightforward. Therefore, several systems tried to combine both deep and shallow linguistic patterns to extract wanted relations. For instance, Xu et al.[36] use HPSG analysis output to construct their patterns. A sample pattern in extracting information about Nobel Prize winners is shown in Figure 2.2

Discovering patterns

The demand to extract information from many different types of relations or events requires pattern-based IE systems to discover patterns automatically. Most studies such as [19] and [24] uses a bootstrapping mechanism to discover more patterns from a seed pattern. The discovery process is an iterative one motivated by the principle of Duality:

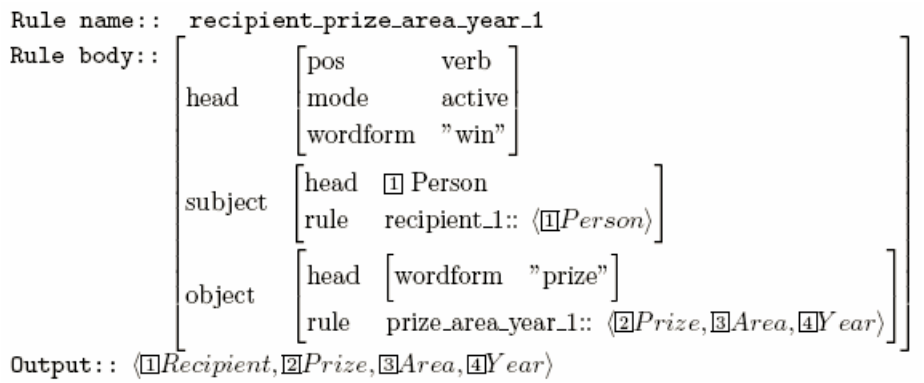

Figure 2.2: Deep linguistic pattern for Nobel Prize event

I. Relevant relation instances are strong indicators of good patterns.

II. Conversely, good patterns are strong indicators of relevant relation instances.

This principle allows us to find new patterns and relation instances, in a bootstrapping fashion. The general algorithm for discovering patterns is:

```
1. Fetch seed instances
2. Extract common patterns from occurrences of these instances
3. Select best patterns.
4. Extract relation instances by using new set of patterns.
5. Filter bad instances.
6. Repeat from step 2 to step 5
```

With this algorithm, previous studies gain success in discovering patterns not only for is-a relations but also for other types of relation such as part-of relations.

2.2 Ontology Extraction

Ontology extraction is a special kind of information extraction. Instead of focusing on some interesting events from users' point of view, ontology extraction aims to build a hierarchy of concepts and the most common relations between them. Studies in ontology extraction mainly focus on extracting a taxonomy from different kinds of resources. A taxonomy is a directed acyclic graph where concepts are linked together

if there is a hypernymy (is-a) relation between them. Because is-a relations are binary relations, common IE techniques might be applied to extract its instances. On the other hand, is-a relations have their own special characteristics, there are several different interesting approaches established for it only such as hierarchical clustering approaches and FCA aproaches. The ontology extraction task can be classified into ontology extraction from text and ontology extraction from other sources of information

2.2.1 Ontology extraction from semi-structured resources

The Internet provides access to an enormous number of valuable resources, many of them are hand crafted. It is possible to generate an ontology by extracting information from existing semi-structure resources.

Ontology from dictionary

Dictionaries are rich sources of detailed semantic information, and are available for many languages. In a dictionary, an entry of a word consists of some information including part-of-speech, synonyms and standard definitions in which a genus of the word usually appears. An average-sized dictionary contains definition for more 10.000 words, this vast amount of information is very useful in constructing a large taxonomy. Work on extracting taxonomy from dictionaries has been started from the seventies with pioneering work by Calzolari (1977) and followed by Amsler (1981)[1], Chodorow et al. (1985)[6], etc. The basic idea is that a genus term tends to be a hypernym of the defined word. Figure 2.3 show some selected definitions in [6] with capitalized genus terms.

From the selected definition, hypernymy pairs such as mean-vehicle, medium-vehicle, vehicle-bicycle, vehicle-tanker can be extracted and form a tangled taxonomy shown in Figure 2.4

A popular method for selecting genus words is head-word extraction. it makes use of shallow analysis of a definition sentence to spot out the head-word in a definition, which is supposed to be the genus word.

```
vehicle: (n) a MEDIUM through which something is expressed, achieved,
             or displayed
vehicle: (n) a MEANS of carrying or transporting something
ambulance: (n) a VEHICLE equipped for transporting wounded, injured,
             or sick persons, or animals
bicycle: (n) a VEHICLE with two wheels tandem, a steering handle, a
             saddle seat, and pedals by which it is propelled
tanker: (n) a cargo BOAT fitted with tanks for carrying liquid in bulk
tanker: (n) a VEHICLE on which a tank is mounted to carry liquids:
            also : a cargo AIRPLANE for transporting fuel
```

Figure 2.3: Selected definitions from a dictionary

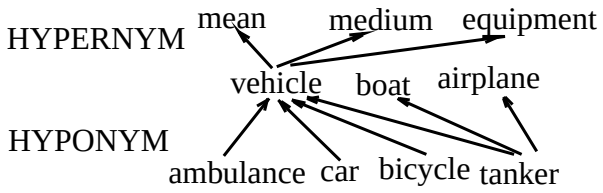

Figure 2.4: A taxonomy extracted from dictionary definitions

Extracting a taxonomy from a dictionary is quite promising, however this method cannot be applied for domains which does not have a dictionary of its own.

Ontology extraction from Wikipedia and WordNet

WordNet is a lexical database developed and maintained by the Cognitive Science Laboratory of Princeton University for English. WordNet groups English words into synsets, sets of synonyms. It provides short, general definitions, and various semantic relations between these synsets. From an ontology point of view, in WordNet, a synset represents a concept or a class. Relations between synsets include hypernymy, meronymy, troponym and some other semantic relations. The hypernymy relation in WordNet spans a directed acyclic graph with a single root node called entity. This graph is a very large taxonomy with 82,115 synsets for 117,798 unique nouns. Although the taxonomy in WordNet is quite reliable, it is too general for a specific domain, it lacks of information about complicated concepts such as "Natural Language

Processing".

Wikipedia is a free, collaborative, multilingual encyclopedia project supported by Wikimedia Foundation. It was launched by Jimmy Wales and Larry Sanger in 2001 and is the most popular and largest general reference work at the moment on the Internet. It has over 3.4 million articles in English which have been written collaboratively by volunteers around the world. Although almost all of its articles can be edited and modified by anyone, its content is highly reliable. Each article in Wikepedia is associated with one or several categories. A category can have subcategories or supercategories. The category system in Wikipedia can be viewed as a directed acyclic graph which highly resembles a taxonomy. Despite this similarity, the subcategory relation in Wikipedia is not exactly an is-a relation, e.g. in Wikipedia, 1980s singer is a subcategory of 1980s music but 1980s singer is not a hyponym of 1980s music.

Ontology extraction from Wikipedia category information has been studied in some work such as [25] and was successfully applied in YAGO. Important information about YAGO can be found in Suchanek et al.[30]. YAGO extracts information about entities from infoboxes in Wikipedia and taxonomy information from the combination of category structure in Wikipedia and WordNet taxonomy. Nevertheless, Wikipedia category structure is not suitable for ontology extraction in specific domain as its resemblance to a taxonomy drop dramatically for specific, uncommon categories.

2.2.2 Ontology learning from text

Ontology extraction from text is popular in domains without other kind of resources. There are quite a lot of studies about extracting hypernymy relations; they can be divided into three groups: pattern-based approaches, clustering based approaches and Formal Concept Analysis (FCA) based approaches.

Pattern-based approach

Like in informatin extraction, pattern-based approaches in ontology learning makes use of linguistic pattern to extract hypernymy relation. The idea was first introduced

by Hearst [16], who used surface pattern to extract explicitly expressed hypernymy pairs. Figure 2.2.2 shows the patterns used by Hearst. Since then, many studies have used Hearst-style patterns in their work on extracting hypernymy relations. These studies include Yang and Callan [37], Mintz et al. [22], Manzano-Macho [12]. Snow et al [28], Fallucchi [13], Snow e[29] used dependency patterns instead of lexical patterns. Some other authors used lexical patterns with additional information such as part of speech, lemma form.

1. NP such as NP
2. such NP as {NP,}* {(and|or)} NP
3. {NP,}* NP {,} and other NP
4. {NP,}* NP {,} or other NP
5. NP {,} including {NP,}* {(and|or)} NP
6. NP {,} especially {NP,}* {(and|or)} NP

Pattern-based approaches have good precision but low recall because these patterns appear rather rarely in the corpus. One more problem with pattern-based approaches is how to build a complete ontology with the extracted relations. Because not all the extracted relations are correct, these approaches usually has to deal with inconsistency. Pattern-based approaches are popular for automatic taxonomy induction. Though suffering from the problems of sparse coverage and inconsistent chains, it is still popular due to its simplicity and high accuracy. To increase the precision and recall, some study applied machine learning with additional information, but linguistic patterns are still the main features. Pattern-based approaches have been applied to extract various types of lexical and semantic relations, including is-a, part-of, synonym.

Clustering-based approach

Clustering

Clustering is a well-studied problem in data-mining and is applied in many practical fields. Given a set of observations, a clustering algorithm groups similar observations into subsets called clusters for further analyses. Observations in a cluster should be similar to each other and different from observations in other clusters.

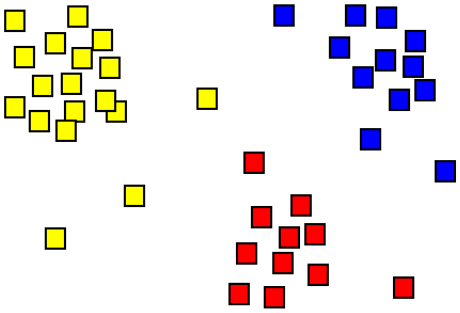

Figure 2.5: Result of clustering (coloring) of the squares into three clusters.

Hierarchical clustering

Hierarchical clustering is one of many algorithms solving a clustering problem. It starts by considering each observation as an individual cluster and then continues merging two most similar clusters into one until the number of clusters is equal to a given number. When the final number of cluster is set to one, the dendogram, a tree which presents the merging process, produced by a hierarchical clustering algorithm resembles a hierarchy of concepts.

Clustering-based ontology learning approaches use clustering algorithms, especially hierarchical clustering algorithms to build a taxonomy. [2], [4], [7] are some of the studies that use a clustering algorithm in ontology learning. This method is often applied for medium-sized corpora because it is able to extract relations that are not explicitly expressed. Clustering-based approaches cluster concepts based on similarities of them. Concepts are usually presented by a vector of features. These features can be contextual features, verb-noun relations, syntactic dependency, co-occurrence, conjunction and appositive features. Experiments show that clustering-based approaches generally fails to produce coherent clusters for small corpora. Moreover, taxonomies produced by clustering-based approaches usually are trees, which are not normally the case in practice where taxonomy is usually a connected acyclic graph. Clustering-based approaches have mostly been applied to extract is-a relations. In

addition, clustering-based approaches have to deal with the challenge of appropriately labeling non-leaf clusters. The labeling amplifies the difficulty in creation and evaluation of taxonomies.

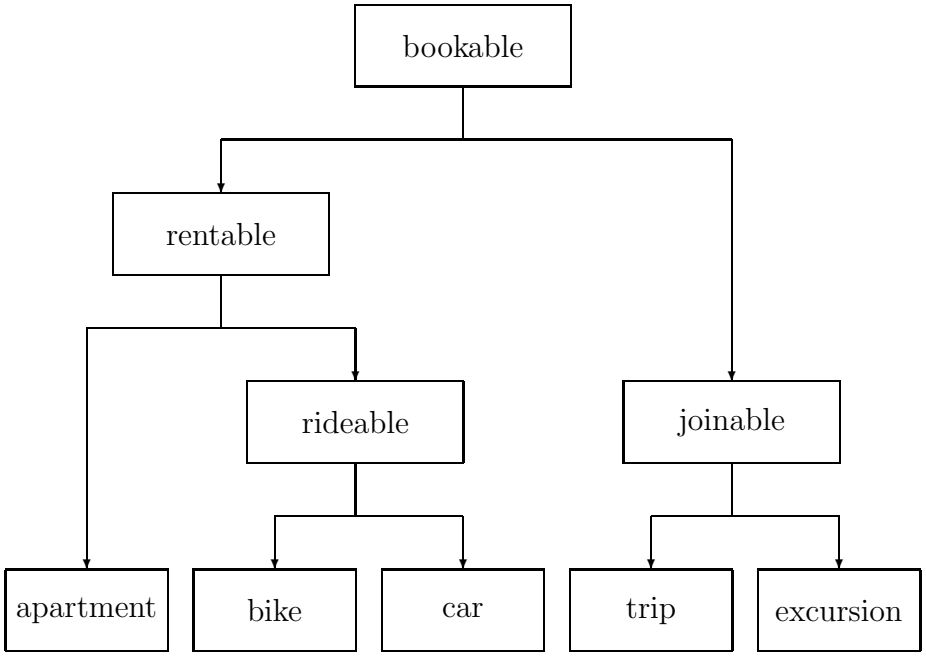

Figure 2.6: Tourist ontology produced by clustering approach

FCA-based approach

Another interesting approach is the FCA-based approach. Formal Concept Analysis (FCA) [15] is a method used for the analysis of concepts, i.e. for deriving relationships between objects and attributes used in describing them. It was introduced in 1982 by Rudolf Wille and has become widely used recently. A formal concept (A,B) in FCA is a combination of a set of attributes B called intend and a set of objects A called extend. In normal cases, given the intend, we can identify the extend and vice versa. The FCA method based on the idea that objects are associated with their attributes, objects with the same attribute should belong to one concept.

The FCA method is adapted and used in ontology learning. These approaches have been addressed in [8], [26], [10], [9], etc. The most used attributes in FCA-based

approaches are contextual information, especially verb-noun interaction. Table 2.1 and Figure 2.7 present a popular example of the FCA method. A set of objects(terms) {hotel, apartment, car, bike, excursion, trip} and a list of verbs that can be used with them are given in table 2.1. FCA-based approaches manage to use this verb-noun interaction as information to build up a taxonomy shown in Figure 2.7

	bookable	rentable	driveable	rideable	joinable
hotel	*				
apartment	*	*			
car	*	*	*		
bike	*	*	*	*	
excursion	*				*
trip	*				*

Table 2.1: Tourist domain knowledge

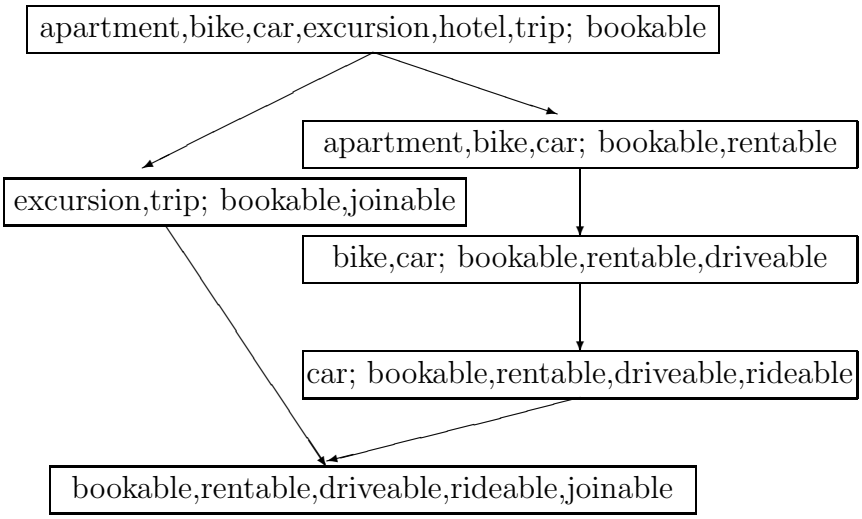

Figure 2.7: Tourist ontology produced by FCA approach

FCA-based approaches also suffer from labeling problems. As shown in the example, the label of a concept is either a set of objects or a set of attributes, not a specific term.

FCA-based approaches still require more research. In practice, its performance is not as good as that of clustering-based or pattern-based approaches as their accuracy is not accecptable for practical use.

Chapter 3

Term extraction

We will present an overview of term extraction in this section. Term extraction forms the basis for our aproach for ontology extraction from text. It is important because it provides us with the terminology in the domain. It helps us to recognize essential concepts whose information needs collecting. There are many techniques for automatic term recognition (ATR), most of them focus on terms that consist of more than one word (multi-word terms) as there is more linguistic and statistical information available to recognize multi-word terms than one-word terms. Even though the number of single-word terms is less than the number of multi-word terms, one-word terms also play an important role in modeling the domain. In an ontology, one-word terms often realize the most basic concepts (e.g. languages, applications), whereas multi-word terms usually represent extended concepts (e.g. agglutinative language, natural language processing application). In order to build an ontology, it is required to extract both multi-word terms and single-word terms. For extracting multi-word terms, we will use the result from applying C-value/NC-value method in Frantzi et al.[14], while for extracting single-word terms, we will use the head noun of multi-word terms as term candidates.

3.1 Multi-word term extraction

Multi-word extraction is a rather well-studied issue. Previously, techniques for multi-word ATR focus on linguistic information; there has been a trend from using only linguistic information to incorporating linguistic and statistical information. Statistical information used in ATR includes frequency of occurrence, mutual information, etc. The method that Frantzi et al. [14] used to extract multi-word terms combines statistical and linguistic information. In this section, we describe their method for multi-word term extraction. Their ATR process is divided into two steps: first, sequences of words that have low potential of being terms are filtered out by the linguistic component, and then the statistical information is applied to select the word chunks which are most likely to be terms. The statistical component consists of two main parts: the C-value and the NC-value. The former aims at improving nested term extraction while the latter aims at improving term extraction in general by incorporating context information.

3.1.1 Linguistic component

The linguistic part consists of the following:

- The linguistic filter applied to the tagged corpus to exclude those strings not required for extraction.
- The stop-list.

The linguistic filter

The role of the linguistic filter is to filter out sequence of words that are unlikely to be terms using linguistic information. In the linguistic component, the corpus needs to be preprocessed by a part-of-speech tagger. Part-of-speech tagging is the process of marking up the words in a text (corpus) as corresponding to a particular part of speech (e.g. noun, adjective, verb, preposition, determiner, etc.), based on both its definition and its context. It is required by the linguistic filter to filter out unwanted sequences of words and permitting only desirable ones for term extraction.

A combination of the below 3 filters is used in the linguistic filter.

- Noun Noun+
- Adj Noun+
- (Adj—Noun)+Noun,

The precision and recall of the term extraction module is affected by the choice of the linguistic filter. A strict linguistic filter will negatively affect recall but will have positive effect on precision. A filter with looser condition, one that permits more types of strings, will decrease precision, but increase recall. For example, a filter which accepts only one pattern: Noun+ Noun will have very good precision, undesirable phrases such as "many compound nouns" would be filtered out. As a contrast, its recall is not sufficient; it is so strict that good terms such as "natural language processing application" are also rejected together with bad phrases.

The combination of 3 filters above could extract more terms than the Noun+ filter, since it extracts not only terms that contain nouns but also terms that contain adjectives. The combination of 3 filters is still a strict filter, it does not permit good terms like "part of speech". In the medicine domain, a filter which accepts strings including preposition may extracts terms like "tetracyclines for ocular rosacea", "scotomas in low vision" or "coloboma of retina", but it also extracts non-terms like "strabismus in children", "composition of tears", "therapy of strabismus", "sensory aspects of strabismus". Nevertheless, to balance precision and recall, the above filter is used since a looser filter will permit many undesirable strings.

The stop-list

In information retrieval, a stop-word is a word that appears many times and does not have an impact on the search process. A stop-list in ATR is a list of stop-words which are not expected to occur as term words. Its use ensures that strings that are unlikely to be terms are not extracted. Some common examples are: "many", "several", "various", etc. A stop-list benefits precision but could leave out terms that contain unexpected words. To remove undesirable sequences of words from the ontology concepts, we used a list of stop-words that includes quantity adjectives, comparative adjectives, and other stop-words such as "similar", "related".

3.1.2 Statistical component

The linguistic information is not sufficient for term extraction. It can only be used to define the structure of a term. Any strings that satisfy some patterns are accepted, but are not necessarily terms. If we use only the linguistic component, many non-terms would be considered as terms. Any noun phrases no matter how unimportant or irrelevant they are in the domain are selected. The statistic information allows recognizing what phrases are important in a particular domain. The C-value defines a measure to calculate the likelihood of a phrase being a term. It assigns a termhood to every candidate string, ranking them in a list of candidate terms, removes strings whose termhood does not exceed a threshold. The measure is built based on statistical characteristics of the candidate string. These are:

- The total frequency of occurrence of the candidate string in the corpus.
- The frequency of the candidate string as part of other longer candidate terms.
- The number of these longer candidate terms.
- The length of the candidate string (in number of words).

Frequency of occurrence is the most used measure in multi-word ATR. It is widely used in early works on term extraction because of the fact that terms tend to occur with higher frequency than non-terms. Frequency of occurrence yields satisfactory results. In works that use frequency of occurrence, termhood is measured by the following formula:

$$termhood(a) = f(a) \tag{3.1}$$

Where
a is the candidate string,
f(a) its frequency of occurrence in the corpus.

The C-value Method

Although frequency of occurrence is a useful statistical measure, it is not sufficient to use only frequency to measure termhood. The first problem is that not every frequently occurring phrase is a term. There are terms whose sub-phrases are not terms. Those sub-phrases occur with a frequency at least equal to that of their super-phrases. One example is "statistical question answering system", the whole phrase is a term in Computational Linguistics, but the sub-phrase "statistical question" is not. Using only frequency, the term extraction module will consider "statistical question" as a term together with "statistical question answering system". C-value is a method that can overcome this problem. In this method, the formula to calculate termhood of a string also considers its frequency as a substring of longer candidate terms.

$$termhood(a) = f(a) - \sum_{b \in T(a)} f(b) \tag{3.2}$$

Where

a is the candidate string,

f(a) is its total frequency of occurrence in the corpus,

T(a) is the set of candidate terms that contain a,

b is such a candidate term,

f(b) is the frequency of the candidate term b that contains a.

A term that occurs inside other terms is called a nested term. The above formulae will only consider a nested phrase as a term if it occurs as a standalone string in a corpus with a significant frequency. Nonetheless, there is still an issue with the formulae. Terms do not have to occur alone in the sentence. There are terms which tend to occur inside other terms, they mostly appear as a nested string inside other term, and rarely appear alone.

If a phrase appears in many longer terms, they also show some "independence" from longer terms they appear in; therefore it is likely to be a term. If a phrase only appears in one term, it is less likely to be a term. The higher the number of its longer terms, the higher the probability of it being a term.

Another parameter which needs considering is the length of the candidate string in

term of number of words. Since a longer string is less probable to appear in a corpus than a shorter string, it provides more information when a longer time appears t times in a corpus than when a shorter term appears t times. Since the maximum length terms cannot be nested in longer terms, and some strings are never found as nested anyway, we distinguish these two cases

- Term candidates which have maximum length or are not a nested term.
- Term candidates which have shorter length and are a part of any longer candidate terms. Their termhood will consider the number of times they occurs as a nested string as well as the number of candidate terms in which they are nested. The former reduces their termhood whereas the latter increases the likelihood of them being independent terms.

The measure of termhood used by Frantzi et al., called C-value is given as

$$C - value(a) = \begin{cases} log_2|a|.f(a) & \text{if a is not nested} \\ log_2|a|.f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b) & \text{otherwise} \end{cases} \tag{3.3}$$

Where
a is the candidate string,
f(.) is its frequency of occurrence in the corpus,
T_a is the set of extracted candidate terms that contain a,
$P(T_a)$ is the number of these candidate terms.

NC-value

Context information

Context information is a common tool in natural language processing. It can be used in word sense disambiguation, synonym finding and other natural language processing tasks. Similarly, context information can also be used in ATR. The basic idea is that extended term units are different in type from extended word units; terms

cannot be freely modified. A common compound noun can be modified by a large number of qualifiers, whereas terms can only be modified by a limited number of qualifiers. Therefore, information about modifiers can be exploited to improve term extraction. In the medical domain, words such as "show", "know", "composed" are usually followed by terms. Frantzi et al. used modifiers from three part-of-speech categories. These are

- nouns
- adjectives
- verbs

In order to apply context information, first, a list of important term context words needs to be extracted from the corpus. Term context words are words that appear in the vicinity of terms. These words are ranked according to the number of terms they appear with. The assumption is that the higher this number, the higher the likelihood that the word is "related" to terms, and that it will occur with other terms in the same corpus.

Frantzi et al. expressed the above criterion with the measure

$$Weight(w) = \frac{t(w)}{n} \tag{3.4}$$

Where

w is the context word (noun, verb or adjective) to be assigned a weight as a term context word,

Weight(w) the assigned weight to the word w,

t(w) the number of terms the word w appears with,

n the total number of terms considered. It is used as a normalization factor.

Because, there is no list of terms available, a list of highest-ranked term candidate is used to extract term context words.

NC-value measure

The final measure for defining termhood is called NC-value measure which combines C-value with context information. The first component is the C-value of the

candidate terms, which has a more important impact; the second component is the context information with smaller impact.

$$NC - value(a) = 0.8 * C - value(a) + 0.2 * \sum_{b \in C_a} (f_a(b) * weight(b)) \tag{3.5}$$

Where
a is the candidate term,
Ca is the set of distinct context words of a,
b is a word from Ca,
fa(b) is the frequency of b as a term context word of a,
weight(b) is the weight of b as a term context word.

The algorithm

The algorithm which combines both C-value and NC-value to extract desirable multi-word terms consists of four steps.

- First step

  Use linguistic filter to select candidate terms.

- Second step

  Apply the C-value method. The output of this process is a list of candidate terms, ordered by their C-value.

- Third step

  Use the highest-ranked term candidates from the second stage to extract of the term context words and their weights.

- Fourth step

  Re-rank the term candidates with NC-value, select candidates whose termhood exceed a threshold.

3.1.3 Output

The C-value and NC-value method extracted 244470 mutli-word terms from the ACL Anthology corpus together with their termhoods. The precision of the output is high. This list of terms provides a starting point for further extraction and refinement.

3.2 One-word term extraction

Single-word extraction is generally considered to be a more complicated task than multi-word extraction. There is less information available that can be used to decide whether a word is a term or not. Frequency, one of the most used measures, is not very useful in this case. A frequently appearing word can either be an important term or a stopword, and a term can occur either frequently or pretty rarely. Our extraction method makes use of the list of multi-word terms. It includes two steps: extracting single-word terms from the set of multi-word terms and extending the set of extracted one-word terms.

We observed that a head noun of a multi-word term tends to be a term. For instance, "language" is the head noun of the term "isolated language" and it is a term. Besides, a single-word term, in general, often is the head noun of some multi-word term. Therefore, being the head noun of a multi-word term is a good indicator of being a term. We decided to extract all the head nouns in the set of multi-word terms to build a set of single-word terms. There are 5000 words extracted from 200000 multi-word terms.

Chapter 4

Building Taxonomy

Taxonomy is one of the main components of an ontology. A taxonomy is an acyclic directed graph representing the is-a relationship between concepts in an ontology. Building the taxonomy is the most important task in ontology construction. It requires collecting concept pairs which have an is-a relationship and constructing an acyclic structure from them.

For building the taxonomy, we propose to use a pattern-based approach combined with applying linguistic information. We choose the pattern-based approach due to its high precision. Works using pattern-based approaches can have a precision of 90%, whereas works using FCA have precisions of around 40% and works using clustering-based approaches have precisions of around 60%. It has a low recall because it cannot extract information that is not explicitly expressed in the corpus, but it has a big advantage over other approaches in terms of precision and simplicity. Although the clustering-based approach and the FCA-based approach can extract information that is not explicitly expressed in the corpus, they produce too many incorrect relations. However the precision of the pattern-based approach is still insufficient to build a reliable taxonomy. Therefore, exploiting linguistic information is necessary to obtain a satisfactory result.

The process of a pattern-based method for building a taxonomy includes

- Finding patterns that express an is-a relationships.

- Collecting is-a pairs by using the discovered patterns.
- Building up a taxonomy from collected pairs.

4.1 Pattern Bootstrapping

Hearst [16] discovered six patterns for taxonomy relations. These patterns are pretty precise but are not sufficient to achieve a high recall. One way to increase the recall of the pattern-based approach is to use bootstrapping to discover more patterns. The more patterns we have, the more information we can extract from the corpus. The idea of bootstrapping has been sketched and used by Hearst in the discovery of the mentioned six patterns. However, the mechanism used in [16] is quite simple. With a more complicated bootstrapping mechanism, we could discover more patterns. Moreover, Hearst conducted her work on a general domain corpus; it is very likely that there are more specific patterns for a scientific domain, in particular our Computational Linguistics domain. By using pattern 1 as a seed pattern, Hearst was able to discover the other 5 patterns. We expect that by applying bootstrapping, we can discover more patterns that representing hyponymy relations. These patterns will be helpful in extracting more hypernymy pairs even if they do not occur as frequently as Hearst patterns.

Because one word can have more than one POS and different corresponding meanings, lexical patterns with POS information are more reliable than lexical patterns with just surface expression. Therefore, we would like to use POS-tagged patterns. A pattern is a POS-tagged string, e.g. "NP such<JJ> as<IN> NP", where a NP is a phrase that matches one of the following regular expressions

```
(DT)? (ADJ | NN)* (NN|NNS) (VBG)?
(DT)? NNP+ (NN|NNS)?
```

There are two main tasks involved in finding patterns: discovering potential patterns and selecting good patterns from the discovered patterns.

4.1.1 Discovering new patterns

The purpose of this step is to generate new patterns from a given list of patterns. The discovery procedure is a bootstrapping process consisting of two main phases: generating hypernymy pairs from the patterns and generating new patterns from the extracted pairs. Normally, in an information extraction system, a pattern discovery procedure starts with one or several seed pairs; in this case, because pattern 1 is a quite precise pattern, we can use the procedure with pattern 1 as a seed pattern instead of using some pairs as seed pairs.

The bootstrapping method used in discovering new patterns is shown in the following procedure.

```
1. Initialize seed pattern such<JJ> as<IN>
2. Put the seed pattern in the set of patterns
3. Find all pairs of noun phrases that match patterns in the
   current set of patterns
4. Select good candidates for hypernymy pairs
5. Find all the strings occuring between the pairs
   together with one or two words on the left or on the right
6. Select the most reliable patterns that do not belong to
   the current set of patterns.
7. Repeat from step 3 until some convergence conditions are met.

The convergence conditions can be:
  - The reliability of the new selected pattern is below a
    threshold
  - The number of loops exceeds a certain value.
```

Figure 4.1: Bootstrapping algorithm for pattern discovery

In the algorithm, the main loop consists of four steps: generating new pairs, selecting good pairs, generating new patterns and selecting good patterns. All these steps will be discussed in detail.

Generating new pairs

This phase takes a set of patterns as input and generates is-a pairs.

When a sentence matches a pattern, we extract the corresponding noun phrase pairs. Two noun phrases in each pair are checked with the term list. Pairs that contain at least one non-term noun phrase are removed because we are only interested in building is-a relation between concepts. The remaining pairs are listed and ranked according to their reliability. Reliability is a measure that considers two factors: the number of times a pair is found and the number of patterns it occurs with. The pairs with highest reliability will be selected and added to the list of pairs used in our bootstrapping process.

To be precise, an additional step is required before comparing the noun phrases extracted with the term list: removing stop-words. Some of the stop-words are "various", "many" and "different". There are several reasons for removing stop-words from the extracted noun phrases. First of all, certain words such as determiners or "very" never appear in the extracted terms and they do not affect the meaning of the whole phrase much, removing them from a noun phrase will increase the recall without harming the precision. Second, some other words do occur in extracted terms but either they are irrelevant or their meaning is dependent on the context (e.g. context-sensitive words such as "similar", "related", "different", or comparative adjectives; quantity adjective such as "many", "several", "various"). Removing these words helps to improve the quality of the ontology.

Selecting good pairs

The reason for selecting only highly reliable pairs lies in the fact that bootstrapping is an easily mislead process. Error generated by one step will propagate to the next steps. Using bad pairs as seeds will generate bad patterns and, as a consequence, generate more bad pairs.

One pair is considered to be good if it is correct and it helps find more patterns. In fact, the latter condition is very hard to justify. Therefore, in practice only the former condition is taken into account. Because there is no automatic way to check whether

a pair is a hyponymy pair, statistical measures are used as a relative justification method. The simplest measure is frequency. A more complicated but more reliable measure takes into account the mutual information of a pair and a pattern.

Generating new patterns

This phase takes a set of is-a pairs as input and generates new patterns as output. Given a pair, first of all, all the sentences that contain both the hypernym and hyponym are extracted. The surrounding context (the strings appearing between them, words on the left and on the right) is considered as a candidate for new patterns. For example: given that "question answering" is a hyponym of "natural language processing", when processing this sentence

```
natural<JJ> language<NN> processing<NN> applications<NNS> like<IN>
question<NN> answering<NN>
```

"like<IN>" will be extracted as one of the candidates for new patterns. All the extracted strings (contexts) will be considered as new pattern candidates. Similar to generated pairs, pattern candidates are also listed and ranked according to their performance. Only the best candidates would be selected to be patterns.

The most important characteristic of a new pattern is reliability. Since not every context string between a hypernymy pair expresses an is-a relation, new patterns produced by bootstrapping methods can easily deviate from the seed patterns. For instance, "English" is a hyponym of "language" (English is a language), the surrounding context of their occurrence in a sentence should express an is-a relation but in the phrase "translating from a number of source languages into English.", the string "into" is not an indicator for is-a relation. If "into" is selected to be a pattern, a lot of false pairs will be selected. Therefore, selecting a new pattern takes a lot of consideration. A good pattern may have even better performance than the seed pattern, whereas a bad pattern can generate a lot of bad pairs that mislead the whole bootstrapping process.

Another desirable characteristic of a new pattern is productivity: a useful pattern should be able to generate an adequate number of new instances. A too specific

pattern can generate good pairs but it may generate only pairs that are already extracted using other patterns. An example of such a pattern is "such as question answering, ", this pattern will only generate pairs similar to (applications, information retrieval), however, all these pairs can be extracted using the "such as" pattern. Therefore, even though "such as question answering" is a reliable pattern, it cannot produce any new patterns. Choosing it as a new pattern is a waste of time and effort.

4.1.2 Ranking patterns

As discussed in the previous section, for the bootstrapping algorithm to work, we need to select good patterns among generated patterns, or in other words, measure the performance of new patterns. The performance measure must take into account two aspects: the productivity and the reliability of the patterns. If the reliability measure is good, we can discover new high-quality patterns, otherwise we may end up finding bad patterns which can mislead the bootstrapping process. The reliability of a new pattern should be directly proportional to the strength of its association with the pairs produced by previous patterns. Pointwise mutual information (pmi) is a commonly used metric for evaluating association strength of two events x and y:

$$pmi(x, y) = \log \frac{P(x, y)}{P(x)P(y)} \tag{4.1}$$

In [24], the reliability of a pattern can be measured by the formula:

$$r_p(p) = \frac{\sum_{i \in I} \left(\frac{pmi(x, y)}{max_{pmi}} * r_i \right)}{|I|} \tag{4.2}$$

Where $r_p(p)$ is the reliability of a pattern,

$r_i(i)$ is the reliability of a relation instance (hypernymy pair),

max_{pmi} is the maximum pointwise mutual information between all patterns and all pairs, it is used for normalization.

The reliability of a relation instance can be computed as:

$$r_i = \frac{\sum_{p \in P} \left(\frac{mi(i,p)}{max_{pmi}} * r_p(p) \right)}{|P|} \tag{4.3}$$

The productivity of a pattern is also important. If only the reliability is considered, the discovered patterns would be too specific. As a consequence, few new pairs or even no new pair will be produced from the newly selected patterns. The productivity of a pattern can be measured by the number of sentences matching it or the number of pairs extracted from it.

$$productivity(p) = \frac{pairnumber(p)}{max_{pairnumber}} \tag{4.4}$$

Where
productivity(p) is the productivity of a pattern p
pairnumber(p) is the number of pairs produced by p
$max_{pairnumber}$ is the maximum number of pairs produced by a pattern.
The final performance of a pattern can be computed as:

$$performance(p) = productivity(p) + r_p(p) \tag{4.5}$$

The generated patterns will be ranked according to their performance. The best pattern is added to the list of patterns and used for extracting new pairs.

4.1.3 Output patterns

After running the bootstrapping process on the corpus, we obtained a number of patterns: some of them are Hearst's patterns. Some other patterns include

4.1.4 Finding verbal patterns

Verbal patterns are the prominent kind of patterns used in pattern-based information extraction. A verb is the main component of a sentence. Characteristics of objects

7. NP ({NP,}* {(and|or)} NP)
8. NP like ({NP,}* {(and|or)} NP)
9. NP {,} e.g. {,} {NP,}* {(and|or)} NP
10. NP (e.g. {,} {NP,}* {(and|or)} NP)
11. NP , except NP

Figure 4.2: Patterns extracted using bootstrapping process

and relations between objects are usually reflected by verbal phrases. At the moment, patterns generated by the automatic bootstrapping process are non-verbal patterns, i.e. patterns that do not contain verbs. The only verb that is exploited to discover hypernymy relation is "to be", however "to be" has many usages (expressing characteristics, etc.) and therefore is not a good indicator for hypernymy relations. If there is another verb that can express is-a relations and almost only is-a relations, it is a good indicator for is-a relations and can be exploited. With verbal patterns, deep linguistic information can be used for better extraction. The goal of this section is to look for verbal patterns that express is-a relations.

In order to find the verb that can replace "to be", we extracted all the verbs that appear between any reliable hypernymy pairs in sentences in the corpus and calculated their frequencies. These verbs were stemmed to their infinitive form; their frequencies are the total frequencies of all the conjugated forms. The result is kind of disappointing. Few words have a frequency comparable to that of "to be" and still their frequencies are less than 1/10 of that of "to be". The verb "include" does appear frequently between hyponymy pairs but mostly in the gerund form, which takes place in a PP, not as the main verb of a sentence. We also expect "to denote" to be a potential verb for "is a" relation but it is used more frequently to express synonym relation or abbreviation.

4.2 Improving patterns

Even though the patterns described in the previous section have high precision, they still extract many wrong pairs. One of the main reasons for this is that lexical patterns have a very low domain of locality. With lexical patterns, we only have the

information about a small context around two noun phrases, the overall structure of the sentences is not considered. Therefore, the actual relationship between phrases in a pattern is not captured precisely in some sentences. A pair extracted from those sentences is not a good one.

An example of this phenomenon is shown in Figure 4.2 which contains the phrase structure of the following sentence

The toolkit should incorporate the major components of an NLP front end, such as a spell checker , a parser and a semantic representation generator.

```
(ROOT
  (S
    (NP (DT The) (NN toolkit))
    (VP (MD should)
      (VP (VB incorporate)
        (NP
          (NP (DT the) (JJ major) (NNS components))
          (PP (IN of)
            (NP (DT an) (NNP NLP) (NN front) (NN end)))
          (, ,)
          (PP (JJ such) (IN as)
            (NP
              (NP (DT a) (NN spell) (NN checker))
              (, ,)
              (NP (DT a) (NN parser))
              (CC and)
              (NP (DT a) (JJ semantic)
                (NN representation) (NN generator)))))))
    (. .)))
```

Figure 4.3: Phrase structure of a complex sentence

In that sentence, “the major components” , not “an NLP front end” is modified by the “such as” PP. As a consequence, the system considers “spell checker” and “major components” as a is-a pair. In fact, there is an is-a relationship between “spell checker” and “NLP front end”.

There are several other types of errors that can happen. It is necessary to improve

the performance of our pattern by adding more knowledge.

4.2.1 PSG analysis

Deeper analysis of a sentence than POS-tagging is required in order to extract relation instances more accurately. With the complete linguistic structure of a sentence, we can filter out cases where lexical patterns do not express the correct relation between noun phrases. However, the deeper the analysis gets, the less precise the produced output becomes. Moreover, deeper analysis also requires more effort. For these reasons, we need to choose a kind of analysis that can give us enough information but no more than that. In complex situations, where exact relationships between a verb and its arguments are important, a deep linguistic analysis such as HPSG analysis is essential. Nevertheless, because our patterns are mostly not verbal patterns, dependency analysis or phrase structure grammar (PSG) analysis would be sufficient. Some works have already used dependency pattern for extracting is-a pair ([28], [29]) and roughly compare between lexical patterns and dependency patterns ([31]). In fact, there is a strong relation between dependency analysis and PSG analysis. One analysis can be more or less constructed from the other. Therefore, we decided to use PSG analysis as a mean to improve the result of the patterns.

Stanford parser

We use Stanford Parser [18] to produce PSG analysis. The Stanford parser is a probablistic parser for natural languages. Probabilistic parsers use knowledge of language obtained from a corpus of hand-parsed sentences such as Penn Tree Bank [21] to try to produce the most probable analysis of new sentences. Despite a few percent of errors, in general, these statistical parser work rather well.

The Stanford parser is a Java implementation of a lexicalized PCFG parser. The Stanford parser for English is trained on Penn Tree Bank corpus. It has a good reputation for parsing performance and is very popular in Computational Linguistic Community.

Figure 4.4 shows the parsing output of a sentence by Stanford Parser in Penn Tree

Bank format.

```
(ROOT
  (S
    (S
      (NP (JJ Other) (NNS modules))
      (VP (VBP do) (RB not)
        (VP (VB require)
          (NP (JJ linguistic) (NN information)))))
    (CC and)
    (S
      (NP (JJ other) (NNS algorithms))
      (VP (VBP are)
        (ADJP (RBR less) (VBN complicated))))
    (. .)))
```

Figure 4.4: An example of the Stanford Parser's parsing output

Benefit of PSG analysis

Using Stanford Parser output, we obtain information about the structure of the whole sentence. The relation between noun phrases can be captured more precisely. We can filter out cases where the patterns do not reflect the real relation between noun phrases.

There are two cases of errors which we can filter out:

- The structure of the sentence is completely different from the structure that a pattern expects
- There is syntactic ambiguity in the sentence and the pattern predicts the structure incorrectly.

Filtering wrong structures

The first type of error with lexical patterns is that they may match totally different structures. Although sentences have different structures underneath, they share some common substring on the surface level. For example:

```
(ROOT
  (S
    (S
      (NP (JJ Other) (NNS modules))
      (VP (VBP do) (RB not)
        (VP (VB require)
          (NP (JJ linguistic) (NN information)))))
    (CC and)
    (S
      (NP (JJ other) (NNS algorithms))
      (VP (VBP are)
        (ADJP (RBR less) (VBN complicated))))
    (. .)))

Other modules do not require linguistic information and other
algorithms are less complicated.
```

Figure 4.5: First example of sentences matching "and other" pattern

Although both sentences in Figure 4.6 and 4.5 match "NP and other NP", only the latter sentence has the right structure to express hyponymy relations: an NP starting with "other" is combined with several other NPs to produce a larger NP . While in the former sentence, the NP starting with "other" is first combined with a VP to make a complete simple sentence.

If only lexical patterns are used, the former sentence will match "NP and other NP" pattern. As a consequence the pair "algorithms", "linguistic information" are extracted and considered as an instance of is-a relation. However, the structure of the sentences indicates that the conjunction "and" connects two simple sentences instead of two noun phrases, and "linguistic information" is not associated with "other algorithms". Therefore "linguistic information" is not an "algorithm".

Errors of this type can be easily filtered out by using the parse tree from Stanford parser. We can construct a syntactic pattern shown in Figure 4.7

Solving ambiguous attachments

A more difficult problem is caused by syntactic ambiguity. Ambiguity is a phenomenon that a sentence can be interpreted in several ways, i.e. it has several possible

```
(ROOT
  (S
    (NP (NNP Acetone))
    (VP (VBZ is)
      (VP (VBN used)
        (S
          (VP (TO to)
            (VP (VB make)
              (NP
                (NP (NN plastic))
                (, ,)
                (NP (NNS fibers))
                (, ,)
                (NP (NNS drugs))
                (, ,)
                (CC and)
                (NP (JJ other) (NNS chemicals))))))))
    (. .)))

Acetone is used to make plastic , fibers , drugs , and
other chemicals .
```

Figure 4.6: Second example of sentences matching "and other" pattern

meaning. Ambiguity can happen on different levels: lexical, syntactic, semantic .

Syntactic ambiguity arises from the relationship between words and phrases in a sentence. When the words and phrases in a sentence can be combined in different ways to form different possible structures, the sentence is said to be syntactically ambiguous.

The most common type of syntactic ambiguity results from the ambiguity in attaching adjuncts. For example: in the sentence "she saw the man with the telescope", the PP phrase "with the telescope" can modify either the noun phrase "the man" or the verb "saw"; this results in different interpretations (meanings) of the sentence.

Lexical patterns for hyponymy relations containing the PP phrases "such as" or "like" also cause this type of ambiguity. Every NP appearing in a sentence before "such as" or "like" has a probability of being modified by them.

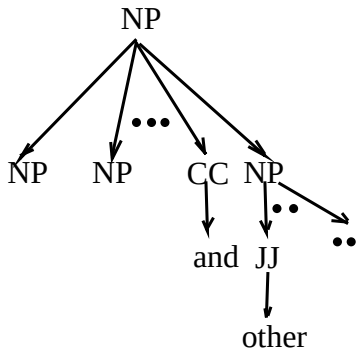

Figure 4.7: "and other" pattern with PSG structure

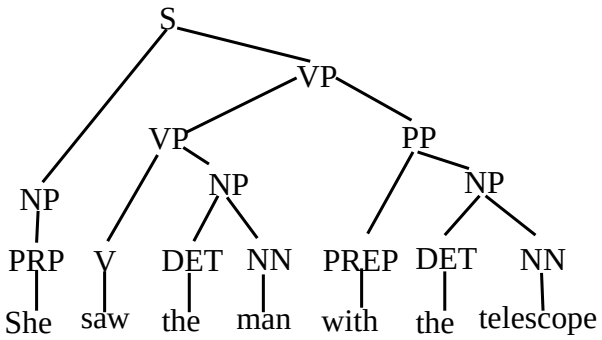

Figure 4.8: Syntactic tree

This is illustrated in the sentences below which are taken from the corpus. All of them are syntactically ambiguous. Noun phrases that can be modified by "such as" are underlined; the noun phrases that are truly modified by "such as" are in bold. It is shown in the examples that the NP modified by "such as" phrase can take many grammatical functions and its position can be anywhere before the "such as" phrase.

- *Therefore,* ***the languages*** *with prefix structure, such as English, German or French, can take the benefits of this direction.*
- *In this section, we briefly review a simple and intuitive approximation method for turning* ***unification-based grammars****, such as HPSG.*
- ***Many existing databases*** *are organized in this type of flat structure, such as Reuters newswire , OHSUMED and TREC .*

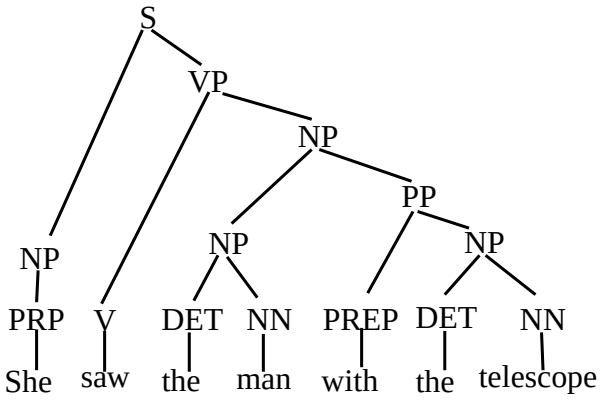

Figure 4.9: Syntactic tree 2

- *Many of the overviewed techniques have been shown to improve solutions to* ***problems*** *that are of particular importance to research scientists, such as homology search and sequence-based functional classification .*

This ambiguity leads to difficulty in selecting the right hypernyms for is-a relation. Normally, a PP is attached to the closest NP, but it can be attached to other NPs with relatively significant probability. Lexical patterns always assume that the PP is attached the closest PP, so they will extract wrong pairs if it is not the case.

Stanford Parser performs pretty well for ambiguous sentences. Among one hundred complicated sentences containing is-a pairs that we extracted from the corpus, using Hearst patterns can only extract fifty correct pairs whereas, the Stanford parsers can extract around than sixty correct pairs. Consequently, it has a positive impact on the performance of our system.

4.2.2 Further improvement

Stanford parser performance for adjunct attaching ambiguity is not bad but is still not sufficiently precise for this task. Decision for this attachment may require more linguistic information or statistical information.

Statistical information

For ambiguous patterns, statistical information can be used to identify one noun-phrase in an is-a pair when the other noun phrase is known. In the "such as" and "like" case, the hyponym is known whereas the hypernym needs to be identified. The idea is to find among the probable noun-phrases a noun-phrase that is most strongly associated with the other noun-phrase, i.e. the noun-phrase having the strongest collocation with the other noun-phrase. There is an abundance of statistical measure to estimate the association between two events, many of them are examined in [17]. The performance of these measures vary in different situations. A measure that works better than others in one case may perform worse in another case. Here are several popular measure:

- Joint Probability.

  $$f(x, y)/N$$

  A statistical measure where the likelihood of two events occurring together is calculated. This measure is quite simple but does not necessarily illustrate the association between two events. Two events which are strongly associated can have a small joint probability if they both rarely happen. Meanwhile, two events which are loosely related can have a big joint probability if they both happen frequently.

- Log likelihood ratio.

  $$\sum f_i j \log \frac{f_{ij}}{\hat{f}_{ij}}$$

  In statistics, a likelihood ratio test is used for comparing two models, to check whether two or more models are related. It can also be used in testing whether two probabilistic functions of two events fit each other, i.e. two events are related.

- Pointwise mutual information(PMI).

$$\log \frac{P(xy)}{P(x)P(y)}$$

  The PMI of a pair of outcomes x and y belonging to discrete random variables quantifies the discrepancy between the probability of their coincidence given their joint distribution versus the probability of their coincidence given only their individual distributions and assuming independence. The measure is symmetric (pmi(x,y) = SI(y,x).) It is zero if X and Y are independent, and equal to -log(p(x)) if X and Y are perfectly associated. It can take on both negative and positive values. Pointwise mutual information is the most used measure for the strength of association. It works well in many circumtances. However, it is less precise for rare events. Two rare events which accidentally occur together are considered to be strongly related by pmi.

- PMI^k.

$$\log \frac{P(xy)^k}{P(x)P(y)}$$

  A family of heuristic association measures defined in (Daille, 1994)[11]. The PMI^k family was proposed in an attempt to investigate how one could improve upon PMI by introducing one or more factors of p(x; y) inside the logarithm.

Because the performance of these measures varies from case to case, we try every measure to see which one of them is the most suitable for our task. Because we do not have testing data for evaluate these measures, we conduct the work on a small set of annotated sentences. Sentences that match one of the patterns are selected and annotated with information about the correct hypernym phrase to form a set of testing data. Values of the measures are still calculated based on the whole corpus. The measure which performs best on this small corpus is used for the whole corpus. This method of choosing the measure is not the best method since the size of the

sample is too small compared to the corpus. The sample may not have the same characteristics as the whole corpus.

Besides, the precision of these measures are affected when one or both of the events occur infrequently. In cases where the hyponym phrase is long, i.e. it rarely occurs (once or twice), using these measures to select the hypernym phrase will favor long phrases, i.e. phrases which occur the least are preferred. To alleviate this effect, we only used the head noun of the hypernym candidates while calculating.

Linguistic information

Linguistic information is also useful in dealing with ambiguity. We can make use of linguistic information in many different levels from syntax to semantics.

Number information

In general, if the head noun of the modified phrase is a countable noun, it should be in plural form, or in singular form with an indefinite article; otherwise, the head noun should be an uncountable noun. Plural form occurs more frequently than other cases. Below are examples for each kind of head noun.

- Plural form: *Therefore, the **languages** with prefix structure, such as English ,German or French ,can take the benefits of this direction.*
- Singular form: *The techniques used by Information Extraction depend greatly on the sublanguage used in a **domain**, such as financial news or medical records.*
- Uncountable noun: *The resulting structures form equivalence classes, since they abstract from word-specific **information**, such as FORM or STEM.*

Moreover, there is a relation between the form of the hypernym phrases and that of the hyponym phrase. This relation is not as strong as subject-verb agreement but it is useful in selecting the right hypernym phrases. When the head noun of the hypernym phrase is in plural form, the hyponym phrase would be either a proper noun phrase or a common noun phrase in plural form, etc.

- *While traditional constituent-based SRL techniques have so far been applied to **languages** characterized by simple morphology and rigid word order, such as **English** and **Chinese**, we think that dependency-based SRL can be particularly useful for languages with a free word order.*
- *A document is a **unit** such as a **sentence** or **paragraph** tagged with a dialogue act.*
- *NPs preceding proper nouns provide **information** such as **occupation** or **affiliation**.*

Cue words

A hypernym phrase usually consists of cue words that expect an illustration. These cue words include:

- words that express an abundance of things (many, a lot, numerous)
- words that express a variety (various, different, other)
- other words (all, each, every)

Among phrases that can be modified by "such as" or "like", the phrase consists of one of these cue words is most likely to be the hypernym phrase.

- *Moreover, it outperforms **other classifiers** for speeding classification such as k-NNFP and k-NN with reduction.*
- *The orthographic expressions of **other categories**, such as nouns, pronouns and numerals, can be generalized to an abstract level by replacing each orthographic expression with a wild card.*

Verb argument structure.

Verb argument structure can also provide us with the capability to disambiguate syntactic ambiguity. There are many sentences having the structure ... V NP Prep NP <our pattern> ... . Such situations are also cases of syntactic ambiguity because both simple noun phrases can be modified by a "such as" adjunct. Verb argument structure can be used to solve part of the problem.

- *This paper describes a project tagging a spontaneous speech corpus with morphological information such as word segmentation and parts-of-speech.*
- *Our parser does not quite reach state-of-the-art performance, even if we limit the competition to deterministic methods such as that of Yamada and Matsumoto.*
- *The toolkit should incorporate the major components of an NLP front end, such as a spell checker, a parser and a semantic representation generator.*

Usually, when the verb in the sentence has the argument list NP Prep NP, the last NP is the hypernym noun phrase. The reason is that the structure NP Prep NP in that sentence is not an NP itself and the writer usually focuses on the last NP.

Information about verb argument can be retrieved from VerbNet [1]. VerbNet is the largest on-line verb lexicon currently available for English. It is a hierarchical domain-independent, broad-coverage verb lexicon with mappings to other lexical resources such as WordNet, Xtag, and FrameNet. VerbNet is organized into verb classes with syntactic and semantic coherence among members of a class. Each verb class in VerbNet is completely described by thematic roles, selectional restrictions on the arguments, and frames consisting of a syntactic description and semantic predicates with a temporal function.

Each Verbnet class contains a set of syntactic descriptions, or syntactic frames, depicting the possible surface realizations of the argument structure for constructions such as transitive, intransitive, prepositional phrases, resultatives, and a large set of diathesis alternations. Semantic restrictions (such as animate, human, organization) are used to constrain the types of thematic roles allowed by the arguments, and further restrictions may be imposed to indicate the syntactic nature of the constituent likely to be associated with the thematic role. Syntactic frames may also be constrained in terms of which prepositions are allowed. Each frame is associated with explicit semantic information, expressed as a conjunction of boolean semantic predicates such as "motion," "contact,' or 'cause."

`http://verbs.colorado.edu/~mpalmer/projects/verbnet.html` is where VerbNet is provided free of charge.

[1] http://verbs.colorado.edu/ mpalmer/projects/verbnet.html

Combining statistical information and linguistic information

To get the best result, we decided to combine statistical information and linguistic information together with the patterns. To combine these sources of information, we use more fine-grained patterns in rule-based fashion.

A rule-based system consists of a list of rules; it is a specific type of knowledge base and a semantic reasoners. A rule-based system produce output by performing the match-resolve-act cycle.

A list of heuristics to select the right hypernym phrase is written in rule-form.

Some of the rules (heuristics) we use are

- If a sentence matches the pattern "NP PreP NP [,] such as" then the NP starting with quantity adjectives (many, numerous) or adjectives that express difference (other, different, various) is the right hypernym phrase.
- If a sentence matches the pattern "NP1 Prep NP2 [,] such as" and the head-noun of NP1 is in (kind, type, number, variety) then NP2 is the probable hypernym phrase.
- If a sentence matches the pattern "V NP1 Prep NP2 [,] such as" and Prep is in the argument list of V, then the second NP is the probable hypernym.
- If a sentences matches the pattern "NP SBAR , such as" then NP is the right hypernym .
- Among all the noun-phrases that occur before "such as" the one that has the highest collocation measure with the child NP is the right hypernym phrase given that their collocation measure is greater than a threshold .

Rules used in our system are not disjoint with each other. One sentence can match with more than one rule. The program will search through all the rules from the start to the end of the program and return the output of the first rule matching with the sentence. Therefore, the order of the rules is very important; it will strongly affect the output hypernym pairs. Generally, if all the sentences matching with a rule r1

also match with a rule r2 (r2 is more general than r1), we put r1 before r2 because there is no point to put r1 after r2 due to the fact that r1 will never be matched with any sentence. However, there are rules that just overlap each other and no rule is more specific. In this case, we put the more precise one above the other.

4.2.3 Ambiguous patterns

Another source of errors is pattern ambiguity. This is caused by the range of meanings of a single word or a phrase, or different ways of using patterns.

"including" can be used to express metonymy relations or hyponymy relations:

- *We describe the ARS as a dag where each relation is a feature structure including three components.*
- *Compounding is common in many languages, including German.*

"is a" is also a generic pattern, which can be used to express many relations.

Ambiguous patterns need to be used more carefully than reliable patterns. When a sentence matches an ambiguous pattern, it is less probably that the extracted noun phrases belongs to an is-a relation. Therefore, the reliability of a pair matching a less reliable pattern should be smaller than that of a pair matching a more reliable pattern. Because of the linguistic and statistical information added, the patterns and their reliability have changed; moreover, the ambiguous pattern "to be" (is a / is an) is not returned by the bootstrapping process, so we do not get the reliability of the "is a/an" pattern. Therefore, we just consider each pattern reliable or unreliable. Pairs extracted by an unreliable pattern are treated differently than those extracted by a reliable pattern. If a pair is extracted by an unreliable pattern, it needs to be verified by another pattern, i.e. it needs to be extracted by another pattern as well.

4.3 Building taxonomy

Given a set of patterns, building a taxonomy simply consists of extracting is-a pairs using the provided patterns and constructing the taxonomy from the extracted pairs.

4.3.1 Extracting is-a pairs

Extracting is-a pairs is only a problem of matching. It is similar to pair extraction in bootstrapping process. Each sentence is compared to the patterns, one by one. If it matches one of the patterns, the heuristics are used to extract the correct pair. The extracted pair is then labeled reliable and unreliable based on the matched pattern.

4.3.2 Constructing taxonomy from extracted pairs

Building a taxonomy from collected pairs is a non-trivial task. Like integrating several sources of information such as databases, combining collected pairs can derive inconsistencies. It happens when there is a cycle in the structure (e.g. "language" is a hypernym of "synthetic language", "synthetic language" is a hypernym of "agglutinative language", and "agglutinative language" is a hypernym of "language). The source of inconsistency can be polysemy of a word, the lack of context information when extracting instances of is-a relation, or the incorrect extraction from syntactic-lexical patterns. These 3 aspects produce undesirable is-a pairs that create inconsistencies. This problem does not occur in FCA approach and clustering approach due to their particular ways of selecting is-a relation instances. To avoid and resolve inconsistencies, we make use of the reliability of selected pairs. Pairs that create inconsistencies are compared against their reliability and the pairs that are less reliable are removed from the taxonomy.

For building a complete taxonomy, we used a simple algorithm to deal with inconsistency.

1. Sort hyponym pairs according to their reliability
2. Take out the most reliable hyponym pair and put it into the taxonomy
3. Check the taxonomy for inconsistency (whether there is a cycle in the taxonomy)
4. Remove the newly added pair if there is inconsistency
5. Move back to step 2 until there is no pair left

Figure 4.10: Algorithm for building taxonomy

Chapter 5

Evaluation

5.1 Game with a purpose

Our method of evaluating the extracted ontology will be discussed in this chapter.

Evaluating an ontology is a non trivial task, especially evaluating an automatically constructed ontology. The difficulty lies in the fact that there is no standard way to model a domain. The frequently used method is to evaluate based on precision and recall. Precision is the fraction of extracted knowledge that correct or reasonable, whereas recall is defined as the amount of knowledge correctly identified with respect to all the knowledge that exists in the corpus. However, both the judgment whether a piece of extracted information is correct and the notion of all knowledge that is described in a text collection are subjective. There are some methods to overcome the problem. The first method attempts to evaluate the quality of constructed ontology by comparing to a "gold standard", which would a well-constructed ontology. The second one is using judgment made by experts who try to examine the accuracy of the extracted ontology. In these two methods, ontology information is divided into several types (terms, taxonomy and other relations) and evaluated seperately. Another method is called application-based evaluation. In this approach, the quality of an ontology is judged based on its practical use in a real-world application.

We decided not to use comparing to a "gold standard" as our evaluation method for two reasons. First of all, there is no gold standard ontology that can be used

as a benchmark for our evaluation since the lack of an ontology in Computational Linguistics is the main reason for constructing an ontology from ACL Anthology. Most of the related works used WordNet as a standard to evaluate the taxonomy. However, WordNet is too general. It is not a ontology dedicated to Computational Linguistics, therefore there are many concepts in Computational Linguistics that are not available in WordNet. Consequently, WordNet is not suitable to be used as a gold standard.

The LT-World ontology is also a candidate for the gold-standard ontology since it is an ontology constructed for Language Technology domain. Even so, LT-world ontology is not a good standard for evaluation. It is created to give information about newest developments, techniques, or competition. Hence it is somewhat news-oriented and does not include needed information. The second reason is: the use of a golden standard ontology for the evaluation of an automatically extracted ontology can lead sometimes to erroneous conclusions regarding the quality of the learned ontology.

A more suitable way to evaluate an automatically built ontology is using human judgment. In general, people who evaluate an ontology should be domain experts. However, collecting domain experts judgment is quite expensive. An alternative option for evaluators would be people who work in the domain, but do not have to know every aspect of the domain. Judgment of one person can be imprecise in some case but this can be overcome by comparing it with judgment of other people. Therefore we would like to collect as many human judgments as possible. Our solution for collecting human judgments is crowdsourcing using GWAP. The method will be discussed in later sections.

The main target to evaluate is is-a pairs, since our multi-word terms are collected using other work's output.

5.1.1 GWAP

GWAP is a concept introduced by Luis von Ahn. A game with a purpose (GWAP) is a computer game that serves some purpose for the people setting up the game by making use of human abilities. Millions of hours are spent each year by human on

playing games. GWAP aims at exploiting that amount of time and effort by creating useful games from which valuable information can be extracted instead of purely entertaining games.

Game with a purpose is a paradigm to make people do things that are trivial to human but complicated for computers through games.

One of the first GWAP is a very interesting game introduced in [34], Verbosity, which serves the purpose of collecting common knowledge. The game requires two players, who alternate being a describer and a guesser. Each turn, the describer is given a secret word. The job of the describer is to help his partner guess the secret word by giving clues. Clues are given in a complete structure decided by game designer. The structures describe characteristics of an object and the describer has to complete the structure with his knowledge. The guesser must type the secret word that his partner is describing. The turn finishes when the guesser enters the secret word correctly.

Another example is the ESP Game [32] which makes image labeling an entertaining effort for humans. It is designed for two players who attempt to assign the same labels to an image. The game collects the results of matches as image labels and the players enjoy the encounter because of the competitive and timed nature of it. To ensure that people do their best to accurately label the images and prevent cheating, the playing partners are chosen at random and unknown to each other; they have only the image in common.

Game with a purpose is used in some other work such as Siorpaes et al. [27], Chamberlain et al. [5].

5.1.2 Evaluation with GWAP

Verifying existence of hyponymy relationship between phrases is a difficult task for computers, but is a simple task for human, although this task requires a certain amount of knowledge when it comes to highly domain specific terminology. If we can design interesting games for people to perform this tasks, we can collect a significant amount of knowledge to evaluate our system performance. By collecting people's

opinion through games, we can get different views of people on the issue, which makes evaluation less subjective then manual evaluation. With a large amount of pairs, creating these games would reduce a substantial amount of effort that otherwise would have to be done by exensive expert.

5.1.3 Designing games

Designing such games is similar to designing an algorithm. It must be correct, efficient, and robust to human diversity in thinking and behavior. We will use the common assumption in GWAP: if many people agree on something, it is more likely to be true. Moreover, according to [33], it is important to keep the games as entertaining as possible because people playing these games would like to have fun instead of doing a task .

Our games are designed based on several principles:

- Number of players: the majority of GWAPs are designed for two people playing at the same time. Points are decided by agreement between players. Players receive good points if they have the same decision. Unlike most of other GWAPs, our games are designed for one player playing at a time. The reason is that our games require players to have a satisfactory knowledge of Computational Linguistics, whereas other games only require common knowledge. Therefore, the number of potential players for our game is much smaller than that of other games. It is very unlikely to have two people playing the same game at the same time. Games for one player are more suitable for our purpose. The players receive more points when their decision is agreed by larger number of players.

- Verifying data: in general, the correctness of information received from players is also decided by majority. If the number of people agreeing with the information is larger than the number of people disagreeing, it is considered to be true. In evaluating hypernymy pairs, a pair is correct if the majority say it is correct.

5.1.4 The games

We build up three games to collect data from players: tetris, invaders, quiz.

Quiz

A quiz is a form of game or mind sport in which the players (as individuals or in teams) attempt to answer questions correctly. Quizzes are also brief assessments used in education and similar fields to measure growth in knowledge, abilities, and/or skills. Although it may be not as fun as other game, Quiz is a very simple game and easy to implement. It is a fast way to gather information.

Invaders

This a shooting game adapted from the popular game: invaders. Each round, players will be given a concept. Groups of four spaceships associated with terms keep on falling down. Players have to shoot the terms which are not hyponyms of the given concept and let the rest fall down.

OntoTetris

This game is similar to a typical Tetris game, but the ground is separated by a wall. Each side of the wall is dedicated for one concept called super-concepts. Each falling piece is also associated with a term called falling concept. The player needs to decide which super-concept the falling concept is a hyponym of. Player will be awarded if his decision is agreed by the majority. Other rules are the same as that of traditional Tetris game.

5.1.5 Evaluate one pair based on player's judgment

For reliability reasons, we mark a pair "correct" if there are at least N people agreeing with it and the number of people agreeing with the pair is larger than the number of

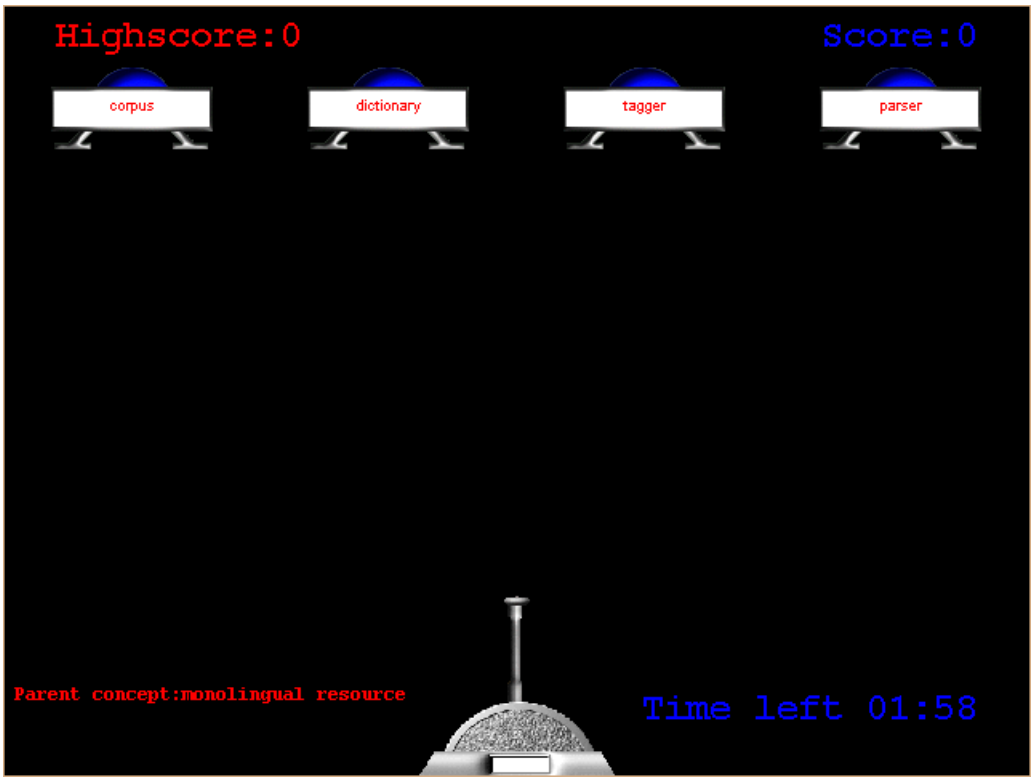

Figure 5.1: Invaders

people who disagree. Similarly, if there are at least N people who do not agree with a pair, it is marked "incorrect". The bigger N is, the more reliable the decision is. N is chosen based on the number of annotated data entered by the players. If we have a large number of annotated data, we will choose a big N.

5.1.6 Sampling result

The sampling method is introduced so that we can evaluate as many is-a pairs as possible within an amount of time. It ensures that even with a small number of players, we can still be able to evaluate all or most of the data. If we randomly select the data presented to players, it will take an incredible long time before there is any pair getting enough judgments to be marked "correct" or "incorrect". We should just focus on a set of pairs at a given time. Therefore, the data presented to the players should be randomly selected from a small set of data needing evaluating. This set of

Figure 5.2: OntoTetris

data is chosen in the beginning; whenever a pair in this set has enough information to be evaluated, it is moved out of the set and another pair is added in. With this set up, we divided the data into three sets: tested, being tested, untested.

Besides, a is-a pair should only be given to a player once; this way, we can speed up the evaluating process. However, it raises a problem when a player keeps on playing until there is no pair left in the "being tested" set that he has not evaluated. To deal with the problem, we added another set which is a reserve set for "being tested" set. When there is no pair in "being tested" set that can be given to a player, if there is some pair in this set that has not been presented to this user, it is selected; otherwise we add another pair from untested set to this set and then give it to the player. The reason for introducing this set is that we should avoid extending the "being tested"

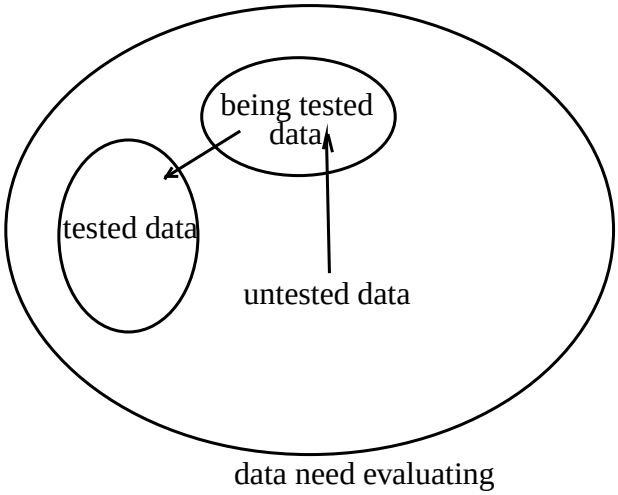

Figure 5.3: Data sampling

set so that we can focus on evaluating just a small set of data.

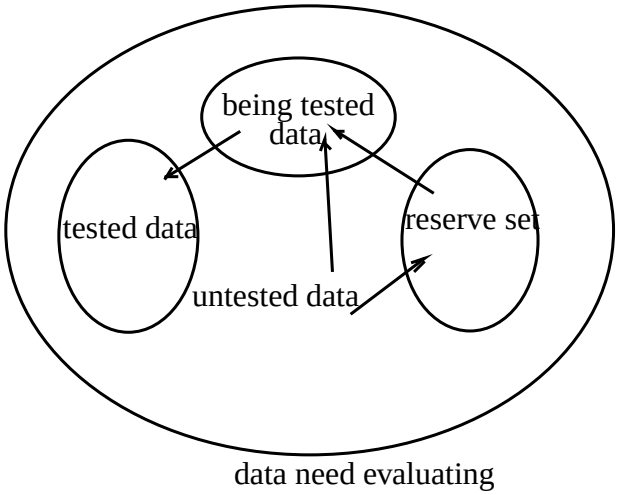

Figure 5.4: Data sampling with a reserve set

5.2 Performance

5.2.1 Game data

We implemented the games in Java. To publicize these games, we converted them into Java applets and put them on a website. This website is deployed using Apache Tomcat, a open source servlet container implementing the Java Servlet and JSP. Tomcat provides a "pure Java" HTTP web server environment for Java code to run.

After the games had been publicized and played by players for ten days, we collected a significant amount of data. There were sixty one voluntary players in total. Nearly seven thousand pieces of annotated pair were collected. Table 5.1 shows general information about the collected data.

statistics	value
number of players	61
number of annotated data	6782
number of pairs shown to players	2940
number of pairs that have 3 people agree	639
number of pairs that have 5 people agree	298

Table 5.1: Annotated data's statistics

5.2.2 Hypernymy pairs evaluation

To evaluate the performance of our systems, we use the precision measure. In this case, precision is the ratio of the number of correct is-a pairs to the total number of the extracted is-a pairs. Because not every extracted pair is checked by the players, the total number of extracted pairs is replaced by the number of pairs which are annotated by the players. Because users can choose whether a pair is correct or not, a pair is said to be correct or incorrect if there are at least N players agree on one choice. Table 5.2 and 5.2 show the precision measure when N is 3 and 5 respectively.

statistics	value
total number of pairs	639
number of correct pairs	490
number of incorrect pairs	149
precision	0.7668

Table 5.2: Performance with N = 3

statistics	value
total number of pairs	298
number of correct pairs	239
number of incorrect pairs	59
precision	0.8020

Table 5.3: Performance with N = 5

The precision of our system is quite good for is-a pair extraction. Works on is-a pair extraction reported precisions from 45% to 90%. However, there is no other research on Computational Linguistics domain so it is hard to compare our system to other systems. Moreover, systems with high precision usually make use of other sources of information. By using only text documents and annotated data collected from the games, we created a number of reliable is-a pairs to build a usable taxonomy.

Because we did not evaluate the result extracted using only Hearst patterns, we also cannot compare the two methods systematically. Nevertheless, among one hundred complicated sentences containing is-a pairs that we extracted from the corpus, using Hearst patterns can only extract fifty correct pairs whereas, our method can extract more than eighty correct pairs.

Chapter 6

Conclusion

This thesis presented an approach for ontology extraction from text and a method for evaluating and improving extracted ontology using Game With A Purpose. Starting with a set of multi-word terms taken form ACL Anthology corpus, we extract a set single-word terms and combine them with multi-word terms to form a complete term list. A set of patterns is initially discovered with bootstrapping method which is then enhanced by adding more linguistic and statistical information. These patterns are used to collect instances of hypernymy relation. A taxonymy is then gradually constructed using the collected hypernym pair. Our method is conducted on a 8000 papers subset of the ACL corpus of Computational Linguistics.

Unlike previous work, we extract both multi-word term concepts and single-word concepts since single-word concepts are basic concepts which play an important role in an ontology. The single-word concepts extracted by our method are meaningful and relevant; they are combined with multi-word terms to form a quite complete set of concepts, hopefully helping the ontology engineer to model the domain in a more solid and compact way. We increase the recall of Hearst pattern by finding new patterns with bootstrapping. We also use statistical and linguistic information to further improve the extracted data.

The GWAP method is applied to exploit human resource in evaluating and further improving the ontology. Using interesting games which are designed to make players justify extracted data, we can save the time and effort to build up a large amount

of reliable ontology information, which is used in both testing and enhancing the ontology. Using the collected data, we were able to evaluate our constructed ontology systematically. The extracted ontology is satisfactory and can be used by other natural language processing applications.

Potential future work includes: integrating the pattern-based approach with other approaches such as FCA or hierarchical clustering. Both FCA and hierarchical approach have a great potential of discovering hidden information. The ontology produced by pattern-based approach still lacks of a lot of information, implicitly expressed information extracted by FCA or hierarchical clustering approach may improve the ontology. It would be interesting to try to distinguish between good information and bad information extracted by FCA or clustering approach.

We could also exploit knowledge from WordNet dictionary and LT-World ontology, in an attempt to enrich the taxonomy extracted from the pattern-based approach. WordNet can be used for relation between general concepts. It is broad-coveraged and is constructed carefully so the taxonomy relationships in WordNet are a valuable resource to exploit. However, there are also some issues in using WordNet. In WordNet, a word has several senses associated with it so it is important to perform word sense disambiguation. Besides, the taxonomy relationships in WordNet are subjective and may conflict with the information extracted from the corpus. Unlike WordNet, LT-World ontology is the source of domain-specific concepts and relationships between them. Nevertheless, exploiting LT-World ontology also takes some consideration and effort, as it is currently mainly modeling the database backbone for the LT-World Web portal.

In addition, we could enrich our ontology with relations other than is-a relation. Part-of relation is another important relation that can be extracted. Besides, other general relations such as synonym, abbreviation and domain-specific relations would make our ontology more complete and usable. The mechanism for extracting taxonomy relationships can be adapted to collect other types of relations. Verbal pattern may be useful in these cases. We can also try other data mining methods, e.g. association rule [20].

Another interesting extension would be distinguishing instances and classes. Is-a

relation is a relation between a class and its superclass as well as relation between an instance and its category. The lexical patterns approach not only extracts subclasses of a class but also extracts set instances (used in [35]). The thesis does not distinguish instances from classes. This distinction is necessary for a fine-grained ontology.

Appendix A

Sample output

Concept	Hyponyms
named entity	location, person, organization, person name, place name, place, organization name, number, protein, gene, capital, number expression, procedure, money, time expression, location name, tax, country, location, rna, company, person, date, date, product name, cell, organization name, time, percentage, proper noun, hmm, proper name, drug, protein name, anatomical site, company name, dna, personal name, city name
grammar	ccg, hpsg, tag, jape, morphological inflection, categorial grammar, lfg, xtag, cfg, translation tool
lexical resource	wordnet, dictionary, thesauri, verbnet, ontology, bilingual dictionary, gazetteer, framenet, encarta, propbank, wikipedia, roget, ltag, eurowordnet, machine-readable dictionary, case frame dictionary, morphological dictionary, monolingual corpus, prop-bank, transfer lexicon, controlled vocabulary, machine readable dictionary, joanis
content word	verb, noun, adjective, adverb, common noun, postpositional particle, temporal adverb, noun phrase, auxiliary verb, human, mouse, verb phrase

Concept	Hyponyms
natural language processing application	information extraction, question answering, machine translation, information retrieval, summarization, parsing, document summarization, speech recognition, statistical machine translation, automatic summarization, pos tagging, named entity recognition, question-answering system, open-domain question-answering, text mining, named entity extraction, automatic lexical acquisition, question-answering, text summarization, document clustering, language model building, word sense disambiguation, annotation projection, cross language information retrieval, natural language parsing, event identification, machine, relation extraction, automatic speech recognition
agglutinative language	Korean, Basque, Chinese, Hungarian, Japanese, Thai
web search engine	Google, Yahoo, Altavista
classifier	svm, decision tree, support vector machine, naive bayes, conditional random field, maximum entropy classifier, dependency path, probabilistic classifier, output score, pruned decision tree, timbl, k-nn, acoustic confidence score
database query language	sql, xpath
vector distance measure	euclidean distance, cosine
dependency relation	subj, subject, object, arg, obj, head-modifier

Concept	Hyponyms
thematic role	theme, agent, instrument, location, patient, experiencer, agent, subject, propbank
open-class word	adjective, adverb, verb, common noun, proper name
agreement feature	gender, number, person, tense
sequence labeling task	named entity recognition, pos tagging, chunking, syntactic chunking
evaluation metric	nist, bleu
morphological feature	number, gender, person, case, aspect, pos, tense, count, voice

Table A.1: Sample output of the systems

Bibliography

[1] Robert A. Amsler. A taxonomy for english nouns and verbs. In *Proceedings of the 19th Annual Meeting of the ACL*, pages 133–138, 1981.

[2] Gilles Bisson, Claire Ndellec, and Dolores Caamero. Designing clustering methods for ontology building: The mo'k workbench. In *In Proceedings of the ECAI Ontology Learning Workshop*, pages 13–19, 2000.

[3] Sergey Brin. Extracting patterns and relations from the world wide web. In *WebDB*, pages 172–183, 1998.

[4] Sharon A. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *In Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 120–126, 1999.

[5] Jon Chamberlain, Massimo Poesio, and Udo Kruschwitz. A demonstration of human computation using the phrase detectives annotation game. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '09, pages 23–24, New York, NY, USA, 2009. ACM.

[6] Martin S. Chodorow, Roy J. Byrd, and George E. Heidorn. Extracting semantic hierarchies from a large on-line dictionary. In *University of Chicago*, pages 299–304, 1985.

[7] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text, 2004.

[8] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence research*, 24:305–339, 2005.

[9] Philipp Cimiano, Andreas Hotho, Gerd Stumme, and Julien Tane. Conceptual knowledge processing with formal concept analysis and ontologies. In *ICFCA*, pages 189–207, 2004.

[10] Philipp Cimiano and Johanna Völker. Text2onto - a framework for ontology learning and data-driven change discovery, 2005.

[11] Béatrice Daille, Éric Gaussier, and Jean-Marc Langé. Towards automatic extraction of monolingual and bilingual terminology. In *Proceedings of the 15th conference on Computational linguistics - Volume 1*, COLING '94, pages 515–521, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[12] Asunción Gómez-Pérez David Manzano-Macho and Daniel Borrajo. Unsupervised and domain independent ontology learning: Combining heterogeneous sources of evidence. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

[13] Francesca Fallucchi and Fabio Massimo Zanzotto. SVD feature selection for probabilistic taxonomy learning. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 66–73, Athens, Greece, March 2009. Association for Computational Linguistics.

[14] Katerina T. Frantzi, Sophia Ananiadou, and Jun ichi Tsujii. The c-value/nc-value method of automatic recognition for multi-word terms. In Christos Nikolaou and Constantine Stephanidis, editors, *ECDL*, volume 1513 of *Lecture Notes in Computer Science*, pages 585–604. Springer, 1998.

[15] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin/Heidelberg, 1999.

[16] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545, 1992.

[17] Hung Huu Hoang, Su Nam Kim, and Min-Yen Kan. A re-examination of lexical association measures. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 31–39, Singapore, August 2009. Association for Computational Linguistics.

[18] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL-2003)*, pages 423–430. Association for Computational Linguistics, 2003.

[19] Dekang Lin and Patrick Pantel. Discovery of inference rules for question answering. *Natural Language Engineering*, 7:343–360, 2001.

[20] Er Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16:72–79, 2001.

[21] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330, 1993.

[22] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[23] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Find your advisor: Robust knowledge gathering from the web. In *WebDB*, 2010.

[24] Patrick Pantel and Marco Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of*

the Association for Computational Linguistics, pages 113–120, Sydney, Australia, July 2006. Association for Computational Linguistics.

[25] Simone Paolo Ponzetto and Michael Strube. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, pages 1440–1445. AAAI Press, 2007.

[26] Guo qiang Zhang, Adam D. Troy, and Keith Bourgoin. 2006a) bootstrapping ontology learning for information retrieval using formal concept analysis and information anchors. In *14th International Conference on Conceptual Structures*, 2006.

[27] Katharina Siorpaes and Martin Hepp. Games with a purpose for the semantic web. 23:50–60, 2008.

[28] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*, 2004.

[29] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 801–808, Sydney, Australia, July 2006. Association for Computational Linguistics.

[30] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA, 2007. ACM Press.

[31] Erik Tjong Kim Sang and Katja Hofmann. Lexical patterns or dependency patterns: Which is better for hypernym extraction? In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 174–182, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[32] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM.

[33] Luis von Ahn and Laura Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.

[34] Luis von Ahn, Mihir Kedia, and Manuel Blum. Verbosity: a game for collecting common-sense facts. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 75–78, New York, NY, USA, 2006. ACM.

[35] Richard C. Wang and William W. Cohen. Automatic set instance extraction using the web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 441–449, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[36] Feiyu Xu, Hans Uszkoreit, and Hong Li. A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In *Proceedings of ACL07, 584–591*, 2007.

[37] Hui Yang and Jamie Callan. A metric-based framework for automatic taxonomy induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 271–279, Suntec, Singapore, August 2009. Association for Computational Linguistics.