Taxonomy Extraction from Wikipedia

A thesis presented to
the faculty of
the Russ College of Engineering and Technology of Ohio University

In partial fulfillment
of the requirements for the degree
Master of Science

Mike Chen
November 2011

© 2011 Mike Chen. All Rights Reserved.
This thesis titled
Taxonomy Extraction from Wikipedia

by
MIKE CHEN

has been approved for
the School of Electrical Engineering and Computer Science
and the Russ College of Engineering and Technology by

Razvan Bunescu
Assistant Professor of Electrical Engineering and Computer Science

__________________________
Dennis Irwin
Dean, Russ College of Engineering and Technology
ABSTRACT

CHEN, MIKE, M.S., November 2011, Computer Science

Taxonomy Extraction from Wikipedia (82 pp.)

Director of Thesis: Razvan Bunescu

Wikipedia is an encyclopedia with a very large coverage of knowledge, that has become an increasingly important resource for extracting structured knowledge.

Much work has been done on extracting taxonomies from Wikipedia. After reviewing previous work, we find that the previous evaluations do not give a clear sense of the systems’ recall. In order to enable a more thorough evaluation, we design an algorithm to generate category subgraphs that are rooted at 10 selected categories from near the top of Wikipedia. The category subgraphs preserve the distribution of the original descendant categories and articles as much as possible by generating random paths in the Wikipedia category graphs that start at the root category and end with a random descendant article. With the exception of the root node in the 10 subgraphs, each node is manually annotated for is-a and instance-of relations with respect to its parent as well as the root node. The newly created datasets enable a more consistent evaluation of taxonomy mining systems.

We also propose a set of relation extraction systems which are designed for two major types of relations: flat relations (node-to-root relations) and hierarchical relations (node-to-parent relations). The taxonomic relation extraction systems are trained and evaluated on the new datasets, exploiting the structure of Wikipedia through a rich set of features. The evaluation on the new datasets gives a clear sense of both the systems’ recall and precision. Thus, in a 10 fold cross validation experiment, we obtain a precision of 89.5% and recall of 88.2% on the task of flat relation extraction. A similar evaluation for hierarchical relation extraction results in a precision of 91.9%, at a recall level of 95.0%.
Approved: 

Razvan Bunescu
Assistant Professor of Electrical Engineering and Computer Science
ACKNOWLEDGMENTS

This thesis would not be complete without the help of my advisor, Dr. Razvan Bunescu. I really appreciate his guidance and support. I also owe many thanks to my committee members, Dr. Cynthia Marling, Dr. Jundong Liu and Dr. Wei Lin. Due to their constructive suggestions, I am able to modify some aspects of the thesis and make it better. I would also like to thank my colleagues Yunfeng Huang and Jincheng Chen and everyone else who helped me with my research. Last but not least, I would especially like to thank my parents for their love and support.
# Table of Contents

Abstract ............................................................................. 3  

Acknowledgments ............................................................... 5  

List of Tables ................................................................. 8  

List of Figures ............................................................... 9  

1 Introduction .................................................................... 10  
   1.1 Motivation ................................................................. 10  
   1.2 Related Work ............................................................ 15  
      1.2.1 Evaluation Methods for Taxonomy Extraction Systems 19  
   1.3 Contributions and Outline ......................................... 20  

2 Subgraph Datasets .......................................................... 22  
   2.1 Dataset Generation ..................................................... 22  
   2.2 Annotation Guidelines ............................................... 24  

3 Taxonomic Relation Extraction and Extraction Features ............... 29  
   3.1 Taxonomic Relation Extraction ................................... 29  
      3.1.1 Flat Taxonomic Relation Extraction ....................... 30  
      3.1.2 Hierarchical Taxonomic Relation Extraction .......... 32  
   3.2 Relation Extraction Features ..................................... 32  

4 System Implementation .................................................... 37  
   4.1 Dataset Generator ....................................................... 44  
   4.2 Information Collector ................................................ 45  
      4.2.1 Parser ................................................................ 45  
      4.2.2 WordNet Information Extractor ......................... 46  
      4.2.3 Mapper ................................................................ 48  
      4.2.4 Disambiguation Phrase Extractor ....................... 49  
      4.2.5 Parent Categories Collector .................................. 49  
      4.2.6 Redirect Pages Collector .................................... 49  
      4.2.7 Infobox Extractor ................................................ 50  
      4.2.8 Coreference Extractor ......................................... 51  
   4.3 Feature Extractor ....................................................... 51  
   4.4 Relation Classifier ..................................................... 54  
      4.4.1 Support Vector Machines .................................... 54  

5 Experimental Results ...................................................... 59
6 Conclusion and Future Work ........................................... 64
  6.1 Future Work ......................................................... 64
    6.1.1 Updating the Datasets ........................................ 64
    6.1.2 Improving the Implementation ............................... 64
    6.1.3 Corpus Based Similarity Measures ........................... 65
    6.1.4 Multilingual Information in Wikipedia ....................... 66
  6.2 Conclusion ......................................................... 66

Appendix: Sample Datasets ............................................. 72
## List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Table Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Dataset generator algorithm</td>
<td>23</td>
</tr>
<tr>
<td>2.2</td>
<td>Depth distribution</td>
<td>24</td>
</tr>
<tr>
<td>3.1</td>
<td>Transitive Closure Classification</td>
<td>31</td>
</tr>
<tr>
<td>4.1</td>
<td>WordNet Information Extractor algorithm</td>
<td>47</td>
</tr>
<tr>
<td>5.1</td>
<td>Extraction performance</td>
<td>61</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1.1 A simple ontology in the animal kingdom. .......................... 10
1.2 Synsets of “actor” in WordNet. .................................. 13
1.3 Hyponyms of “actor” in WordNet. ............................ 14
1.4 Hypernyms of “actor” in WordNet. ............................. 15
1.5 A snapshot of the category “corporations” in Wikipedia. .......... 16
1.6 The infobox of “Clamp (manga artists)” ....................... 17

2.1 The 10 root categories in the dataset. ............................ 23
2.2 Sample annotations for Stars. .................................. 26

4.1 The Dataset Generator of the Taxonomic Relation Extraction Systems. ..... 37
4.2 The training part of the Taxonomic Relation Extraction Systems. ........ 38
4.3 The testing part of the Taxonomic Relation Extraction Systems. .......... 39
4.4 An example of input category subgraphs. .......................... 40
4.5 Structure and relations between different components of information collector. 41
4.6 The true article page of the category “Artisans.” .................. 43
4.7 The redirect pages of “Science Museum Oklahoma.” ................ 50
4.8 An example for Feature Extractor. ................................ 52
4.9 Illustration of k-fold cross validation. ............................. 53
4.10 The Margin in the Max-Margin Classifier. ...................... 54

5.1 ROC graph for flat taxonomic relation extraction. .................. 60
5.2 ROC graph for hierarchical taxonomic relation extraction. ........... 61

A.1 A Portion of the Dataset Artists. ............................... 73
A.2 A Portion of the Dataset Awards. ............................... 74
A.3 A Portion of the Dataset Beverages. ............................ 75
A.4 A Portion of the Dataset Corporations. ........................ 76
A.5 A Portion of the Dataset Films. ................................. 77
A.6 A Portion of the Dataset Natural disasters. ..................... 78
A.7 A Portion of the Dataset Restaurants. .......................... 79
A.8 A Portion of the Dataset Sports clubs. .......................... 80
A.9 A Portion of the Dataset Stars. ................................. 81
A.10 A Portion of the Dataset Viruses. .............................. 82
1 INTRODUCTION

1.1 Motivation

Ontology is a branch of philosophy that is concerned with the creation of a systematic way of describing existence, for example by classifying all the entities in the universe into a hierarchy of fundamental categories (Ross, 1924). Artificial Intelligence (AI) researchers and practitioners think of an ontology as a formal knowledge representation that describes all of the relevant things in a domain and the ways in which these things are related to each other. Figure 1.1 shows a simple ontology that contains relevant entities in the animal kingdom and the ways in which they are related to each other. For example, a “herbivore” is a kind of “animal,” therefore they are in an is-a relation, while “Columbus zoo” is an instance of “zoo,” so they are in an instance-of relation. A taxonomy may be viewed as an ontology in which all of the relationships are of the type is-a or instance-of.

In the 1970s, computer scientists realized that many AI tasks can benefit from ontologies of everyday common sense knowledge (Copeland and Proudfoot, 2007). So an
increasing number of ontologies have been developed since then to provide much-needed structured knowledge to AI applications, especially in Natural Language Processing (NLP). Structured knowledge plays an increasingly important role in many NLP tasks, such as word sense disambiguation (WSD), machine translation (MT), name entity recognition (NER), question answering (QA), text classification, and Information Retrieval (IR). This structured knowledge can be extracted from repositories such as Wikipedia, WordNet and Cyc. Wikipedia is a free, multilingual encyclopedia written collaboratively by people all over the world (Giles, 2005). WordNet is a lexical database used for many NLP tasks, the basic structure of which is the synset (Miller et al., 1990). Cyc is an AI project which assembles an ontology of everyday common sense knowledge (Lenat and Guha, 1989).

Bunescu and Pasca (2006) use information from Wikipedia to solve the problem of name entity disambiguation. First a dictionary D of named entities is built from Wikipedia. Then a method for name entity disambiguation is designed, which takes as input a proper name in its document context, detects whether the proper name refers to an entity from D as well as finds the exact Wikipedia entity referred in that context. Niladri Chatterjee and Naithani (2005) focus on the task of machine translation. WordNet is used to extract semantic information about the sentence being translated and the components of the sentence, which helps to solve the pattern ambiguity problem as well as simplifies the task of translating sentences correctly into the target language. In (Cohen and Sarawagi, 2004), an external named entity dictionary is used to improve name entity recognition.

---

1. Word sense disambiguation (WSD) is a NLP task which solves the following problem: Given a word with several meanings, it identifies the word sense being used, according to the context (Navigli, 2009).

2. Named entity recognition (NER) is a NLP task which identifies and classifies noun phrases, text spans and sequence of words in documents into predefined categories. These predefined categories may include persons’ names, organizations’ names, locations’ names and so on (Poibeau and Kosseim, 2001).


4. A synset is a set of synonyms which contains an underlying meaning.

5. When translating English to Hindi, two English sentences which have similar structures may generate two Hindi sentences with very different structures. So another type of ambiguity, called pattern ambiguity, is created, which refers to the difficulty in deciding the structure of the translation instead of the meaning (Niladri Chatterjee and Naithani, 2005).
systems by incorporating similarities between the extracted entities and the entries in the dictionary. Warren A. Hunt and Nyberg (2004) argue that the availability of rich resources will be increasingly critical to the QA performance and proceed to quantify and bound the potential impact of different on-line resources such as the Web, WordNet, gazetteers, and encyclopedias on the QA performance. In (Ifrim and Weikum, 2006), the accuracy of text classification is improved by using ontological knowledge bases. Liu et al. (2004) disambiguate word senses by using WordNet and apply their word sense disambiguation methods to the task of document retrieval.

The need for structured knowledge has led researchers to work on taxonomic and ontological knowledge resources. Over the past decades, a number of knowledge repositories were developed. Some of them are manually assembled knowledge bases, such as Cyc and WordNet. Others, such as Wikipedia, are edited collaboratively by people all over the world.

Cyc (Lenat and Guha, 1989) is an AI project which intends to assemble an ontology of common sense knowledge in order to make AI applications accomplish human-like reasoning. However it receives criticisms because the current system is incomplete in both breadth and depth (Bertino et al., 2001).

Another man-made knowledge repository is WordNet (Miller et al., 1990). WordNet is an English lexical database that provides a comprehensive coverage of word senses by grouping English words into synsets which are represented as sets of synonyms. Each synset is also associated with a short definition called gloss.

Furthermore, WordNet specifies a diverse set of semantic relations between synsets. The goal of WordNet is twofold: to provide a dictionary as well as to enable automatic text analysis and to support NLP applications.

The semantic relations that connect most synsets to other synsets are very important components of WordNet, especially the “hypernym” relation and its semantic opposite,
“hyponym”. A “hypernym” is a generalization of a concept, for example, “animal” is a
hypernym of “cat,” because every “cat” is a (kind of) “animal.” There are two types of
hypernym relations: \textit{is-a} relations and \textit{instance-of} relations.

Figures 1.2 to 1.4 show a sample of structures from WordNet. However, WordNet has
its own problems and limitations. For example, WordNet only contains a few thousand
named entities. Another limitation of WordNet is that although it has a wide coverage of
common concepts, it does not contain vocabularies for special domains.

Due to the problems and limitations of manually assembled ontologies, researchers
have recently directed a substantial amount of effort to extracting ontologies from
Wikipedia. The most notable advantage of Wikipedia is its large coverage of knowledge,
in particular at the instance level, which is one of the most important features of knowledge.
bases for NLP applications (Ponzetto and Strube, 2007). It not only has a wide coverage of
cryptic entries, but it also features a comprehensive domain orientation in its category
system (Ponzetto and Strube, 2007). However, the Wikipedia categories are thematically
organized and therefore the category system lacks a fully fledged subsumption hierarchy.
This means that Wikipedia still lacks the kind of semantic structure that would allow a
computer system to reach a human level understanding of natural language.

As shown in Figure 1.5, Wikipedia is only a semi-structured taxonomy, i.e. even
though “Corporate crime” is listed as a subcategory of “Corporations”, it is not really in a
subsumption relationship with “Corporations”. However, since Wikipedia contains many
subsumption relations, researchers have designed methods for exploiting the structure of
Wikipedia in order to automatically extract taxonomies from Wikipedia.
Figure 1.6 shows a snapshot of the Wikipedia article “Clamp (manga artists).” The figure shows the definition paragraph, the coreference words of the title, and the infobox, all of which could provide useful discriminative features for the extraction of ontological relations from Wikipedia.

Knowledge bases that are extracted automatically from Wikipedia, are usually orders of magnitude larger than hand-crafted resources. However, the proposed methods, being automatic, are not 100% accurate and they might miss true subsumption relations. Consequently, it is important to estimate how many true subsumption relations they extract from Wikipedia.

1.2 Related Work

A large body of work concerned with extracting knowledge structures automatically from Wikipedia has been done recently.
The YAGO project links facts which are derived from Wikipedia to concepts in WordNet with an estimated accuracy of 97%. YAGO not only builds the is-a hierarchy but also extracts non-taxonomic relations. A combination of rule-based and heuristic methods is introduced. The resulting knowledge base is superior to WordNet both in quality and quantity: The newly generated resource adds knowledge about individuals with their semantic relationships and increases the number of instances by several orders of magnitude (Suchanek et al., 2007).
In (Ponzetto and Strube, 2007), the Wikipedia category system is taken as the input, and a series of methods which are based on connectivity in the category system and lexico-
syntactic patterns are used to annotate the relations between categories. As a result, a large scale taxonomy, called WikiTaxonomy, is derived from the Wikipedia category system.

In (Nastase and Strube, 2008), an approach is introduced to acquire knowledge from Wikipedia categories by means of decoding the names and referring back to the network to induce relations between concepts in Wikipedia. This research complements the type of work in which the category network is transformed into a taxonomy. More semantic relations, such as located-at and part-of, are generated by this approach.

KOG (Wu and Weld, 2008) is an automatic system that builds a rich ontology by combining Wikipedia infoboxes and WordNet concepts. The KOG system relies on Wikipedia infoboxes, whose rich collections of attributes provide a diverse and substantial set of ontological relations.

Ponzetto andNavigli (2009) build a system that disambiguates categories and restructures the previous WikiTaxonomy by using a two-phase methodology. The approach starts by mapping WikiTaxonomy to WordNet. To obtain the mapping, the project utilizes a knowledge-rich method maximizing the overlap between the structure of the source knowledge and the structure of the target knowledge. The resulting mapping is then used to restructure the Wikipedia taxonomy. Restructuring steps are made on the Wikipedia categories that have the highest degree of inconsistency when compared with the corresponding WordNet hierarchy. Linking WikiTaxonomy to WordNet makes the Wikipedia taxonomy structure better comply with a man-made knowledge resource.

DBpedia (Bizer et al., 2009) is another project that extracts structured information from the Wikipedia knowledge base. This knowledge base covers many domains and evolves automatically as Wikipedia changes. The DBpedia project uses four classification

6 An infobox is a table on the top right-hand corner of a Wikipedia article, which presents the values of some common attributes of the entity described in the article (see Figure 1.6 for an example).
schemata to organize entities: Wikipedia Categories, YAGO, UMBEL\textsuperscript{7} and the DBpedia Ontology.

The MENTA project (de Melo and Weikum, 2010) explores the multilingual nature of Wikipedia. It is able to form a taxonomy by integrating not only WordNet synsets but also entities from all language editions of Wikipedia. At the heart of MENTA is an algorithm that first links Wikipedia pages and infoboxes to synsets in WordNet using a set of heuristic functions. The algorithm then maps entities to their parent entities or equivalent entities.

1.2.1 Evaluation Methods for Taxonomy Extraction Systems

The Wikipedia category graph lacks many of the basic principles which are used in manually constructed knowledge bases, especially for the more abstract concepts that usually appear in upper level ontologies. Hence, a recurrent effort in most ontology extraction methods has been made to link the extracted ontology to hand-crafted knowledge bases such as WordNet and Cyc. The evaluation of the taxonomic relations generated from Wikipedia is often done by sampling from top-ranked extracted relations or by comparing against the linked knowledge bases.

In order to evaluate the performance of the YAGO system, facts are randomly selected from the extracted ontology. Then anonymous human judges are asked to evaluate whether the extracted relation pairs are correct. The ground truth of Wikipedia is also used to complement the fact that common sense is not always sufficient to assess the validity of an extracted pair (Suchanek et al., 2007).

In (Ponzetto and Strube, 2007), in order to evaluate the extracted taxonomic relations, first for each category pair, each category is mapped to its Cyc concept. Only 85\% of the pairs which have corresponding concepts in Cyc are evaluated. These pairs are evaluated

\textsuperscript{7} The full name is Upper Mapping and Binding Exchange Layer, and it is a subset extracted from OpenCyc.
by querying Cyc whether the concept denoted by the Wikipedia subcategory is either an instance of or is generalized by the concept denoted by its parent category.

For the purpose of evaluating the precision and recall of subsumption detection in KOG, Wu and Weld (2008) align their ontology to WordNet synsets through the use of DBpediaMap, a manual mapping of 290K articles to corresponding WordNet nodes provided by DBpedia. To train and evaluate the extracted ontology, they use a pseudo-dataset derived from DBpediaMap.

In (Ponzetto andNavigli, 2009), an indirect assessment of the restructured taxonomy is obtained by manually evaluating a random set of 200 restructuring steps.

To evaluate the accuracy of the MENTA taxonomy (de Melo and Weikum, 2010), a test set of 234 mappings are randomly selected and manually assessed. In order to better evaluate the performance of the system, an algorithm is designed to select mappings with high confidence. Another assessment is done on over 100 highly confident mappings.

1.3 Contributions and Outline

While the evaluation of the taxonomic relations by sampling from top-ranked extractions or by comparing with another resource is appropriate for estimating the precision, such an evaluation can not give a clear sense of the system’s recall\textsuperscript{8}. In this thesis, a dataset of category subgraphs which are automatically sampled from the Wikipedia category graph is introduced. In this dataset, the node-to-parent and node-to-root pairs are manually labeled with is-a and instance-of relations. The new datasets can be used as benchmark datasets to evaluate the precision as well as the recall of any proposed model for taxonomic relation extraction. The newly created datasets can also be used to train relation extraction systems. A set of learning approaches are designed for two major types of taxonomic relation extraction: flat extraction of node-to-root relations,

\textsuperscript{8} The recall is computed by dividing the number of samples which are correctly classified as positive (true positives) by the number of samples which are labeled as positive (Makhoul et al., 1999).
and hierarchical extraction of node-to-parent relations. These approaches aim to train a relation extraction system to identify which of the pairs in the Wikipedia category graph are positive subsumption relations. Hence, the trained system takes a Wikipedia category graph as input, exploits the structure of Wikipedia and WordNet through a set of features and automatically classifies pairs of node-to-root relations or pairs of node-to-parent relations. Experimental evaluation shows that the best performing flat relation system obtains an accuracy of 85.2%, whereas the best performing hierarchical relation system obtains an accuracy of 88.7%.

The remainder of this thesis is organized in the following way. Chapter 2 introduces the newly created dataset of category subgraphs which are automatically sampled from the Wikipedia category graph. In Chapter 3, a set of learning approaches for flat and hierarchical relation extraction are described along with the underlying extraction features. The implementation of the system is explained in Chapter 4. Chapter 5 presents the experimental evaluation setting and a discussion of the results. We conclude with suggestions for future work in Chapter 6.
2 SUBGRAPH DATASETS

2.1 Dataset Generation

The set of articles and categories in Wikipedia form a directed acyclic graph in which inner nodes are categories and outer nodes are articles. We selected 10 diverse categories (shown in Figure 2.1) from near the top of the hierarchy as root nodes for the dataset generator algorithm shown in Table 2.1. Since manually annotating all the descendant categories and articles under the 10 root categories is unfeasible, the purpose of this algorithm is to generate a subgraph for each root category that preserves as much as possible the distribution of the original descendant categories and articles. The algorithm generates random paths in the Wikipedia category graph that start at the root category node $R$ and end with a random descendant article. Each path is initialized with the root category, and at each step its length grows by appending to it a random node selected from the current node’s children $N,C$. Then the random node becomes the current node and the path terminates whenever the current node has no children (i.e. it is an article node). The overall path generation process stops when the set $P,A$ of article nodes contained in the generated paths reaches a predefined size $S = 200$. Due to the fact that an article or category node may have multiple parent categories, two or more different paths may terminate at the same article, which means that the number of randomly generated paths may be strictly greater than the number of articles ending the paths. The random paths thus generated correspond to a subgraph of the original Wikipedia\(^9\) category graph rooted at category $R$. The actual implementation of the algorithm in Table 2.1 is further optimized to avoid considering nodes whose descendants have all been selected in the previous paths, and consequently to stop when all descendant articles have been selected (to cover the unlikely cases when a root category has fewer than $S$ descendant articles). Articles and categories that are marked

\(^9\) We used the January 16, 2010 Wikipedia xml dump.
Artists (215, 183, 204), Awards (221, 136, 174), Beverages (417, 261, 339), Corporations (260, 104, 198), Films (443, 351, 406), Natural disasters (260, 135, 176), Restaurants (275, 237, 257), Sports clubs (339, 313, 323), Stars (190, 132, 161), Viruses (252, 56, 193)

Total (2872, 1908, 2431)

Figure 2.1: The 10 root categories in the dataset.

as problematic (e.g. cleanup) or incomplete (e.g. stub) are automatically filtered out from the generated dataset.

Table 2.1: Dataset generator algorithm.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>set $\mathcal{P} \leftarrow {}$</td>
</tr>
</tbody>
</table>
| 2.   | while $|\mathcal{P}.\mathcal{A}| < S$:
| 3.   | set $N \leftarrow R, \ path \leftarrow [R]$ |
| 4.   | while $N$ is a WP category:
| 5.   | set $N \leftarrow$ a random child node from $N.C$ |
| 6.   | set $\ path \leftarrow [path|N]$ |
| 7.   | if $\ path \not\in \mathcal{P}$:
| 8.   | set $\mathcal{P} \leftarrow \mathcal{P} \cup \{\ path\}$ |
| 0.   | return $\mathcal{P}$, a directed acyclic graph rooted at $R$ |

Using the algorithm in Table 2.1 we generated 10 subgraphs, one for each of the 10 root categories. Table 2.1 below shows the overall distribution of nodes with respect to
their depth in their respective subgraph. Since each subgraph is a directed acyclic graph, for any given node its depth was considered to be the length of the shortest path between that node and the root category.

<table>
<thead>
<tr>
<th>$d=1$</th>
<th>$d=2$</th>
<th>$d=3$</th>
<th>$d=4$</th>
<th>$d=5$</th>
<th>$d=6$</th>
<th>$d=7$</th>
<th>$d\geq 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>249</td>
<td>958</td>
<td>894</td>
<td>536</td>
<td>289</td>
<td>78</td>
<td>28</td>
<td>12</td>
</tr>
</tbody>
</table>

### Table 2.2: Depth distribution.

#### 2.2 Annotation Guidelines

With the exception of the root category node, each node in the 10 subgraphs was annotated for *is-a* and *instance-of* relations with respect to both its parents and the root node. If a node $N$ has $k$ parent categories $P_1$, $P_2$, ..., $P_k$ in the subgraph rooted at $R$, then the $k+1$ pairs $(N, R)$, $(N, P_1)$, $(N, P_2)$, ..., $(N, P_k)$ were manually annotated, using the following set of annotation labels:

- **C**: A pair of nodes $(N_1, N_2)$ is labeled as $C$ iff $N_1$ is a subcategory of $N_2$. This is analogous to the hypernym *is-a* relations in WordNet. Most of the $C$ labeled pairs consist of category names, such as (Port authorities, Corporations).

- **I**: A pair of nodes $(N_1, N_2)$ is labeled as $I$ iff $N_1$ is an instance of category $N_2$. This is analogous to the hypernym *instance-of* relations in WordNet. Most of the $I$ labeled pairs consist of an article title and a category name, such as (BBC, Corporations).

- **LI**: A pair of nodes $(N_1, N_2)$ is labeled as $LI$ iff $N_1$ is a list of instances of $N_2$. The first node in most of the $LI$ labeled pairs has a title that starts with *List of*, as in (List of stars in Pavo, Stars).
• **LLI**: A pair of nodes \((N_1, N_2)\) is labeled as **LLI** iff \(N_1\) is a list of lists of instances of \(N_2\). The first node in most of the **LLI** labeled pairs has a title that starts with *Lists of*, as in *(Lists of films, Films)*.

• **SI**: A pair of nodes \((N_1, N_2)\) is labeled as **SI** iff \(N_1\) is a system of instances from \(N_2\), such as *(Kappa Piscium, Stars)*. We use *system* here in a very general sense of the word – in this example *Kappa Piscium* is a multiple star, which is a complex of three or more stars.

• **TC**: A pair of nodes \((N_1, N_2)\) is labeled as **TC** iff \(N_1\) is a category for types of instances of \(N_2\). The first node in most of the **TC** labeled pairs has a title that ends in *types of* or that starts with *Types of*, as in *(Types of restaurants, Restaurants)*.

• **BY**: A pair of nodes \((N_1, N_2)\) is labeled as **BY** iff \(N_1\) is a meta-category for types of instances of \(N_2\), as in *(Films by culture, Films)*.

• **O**: This is a catch-all label for node pairs not in any of the previous categories.

Figure 2.2 shows a few example annotations from the subgraph rooted at the category *Stars*. Each line contains the node title, the Wikipedia namespace number (14 for categories, 0 for articles), and the label with respect to the root, followed by the label with respect to the parent.

One important principle that was followed during annotation was that the information in Wikipedia takes precedence over information from any other sources and therefore the annotation of subsumption relations should be consistent with Wikipedia. A supernova, for example, is defined in Wikipedia as “a stellar *explosion* that is more energetic than a nova”\(^{10}\), whereas in WordNet its gloss is “a *star* that explodes and becomes extremely

luminous in the process”. Because a stellar explosion is not really a type of star, giving precedence to Wikipedia meant that *Supernova* was tagged as *O* with respect to the root category *Stars*. The meaning of a category title was determined based on the following information, in this order: the main article associated with the category (whenever present), the description text associated with the category (whenever present), and lastly the category title itself. The meaning of an article title was determined based on the text of the article. In the rare instances when the text was not sufficient to properly disambiguate the meaning

<table>
<thead>
<tr>
<th>Stars 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable stars 14 <em>C C</em></td>
</tr>
<tr>
<td>Kappa Piscium 0 <em>SI O</em></td>
</tr>
<tr>
<td>Gliese 581 0 <em>I I</em></td>
</tr>
<tr>
<td>Peculiar variables 14 <em>C C</em></td>
</tr>
<tr>
<td>V838 Monocerotis 0 <em>I I</em></td>
</tr>
<tr>
<td>Supernova impostor 0 <em>O O</em></td>
</tr>
<tr>
<td>Eruptive variables 14 <em>C C</em></td>
</tr>
<tr>
<td>Coronae Borealis variables 14 <em>C C</em></td>
</tr>
<tr>
<td>DY Persei variables 14 <em>C C</em></td>
</tr>
<tr>
<td>Stellar deities 14 <em>O O</em></td>
</tr>
<tr>
<td>Stellar gods 14 <em>O C</em></td>
</tr>
<tr>
<td>Hesperus 0 <em>O I</em></td>
</tr>
<tr>
<td>Stellar goddesses 14 <em>O C</em></td>
</tr>
<tr>
<td>Atarsamain 0 <em>O I</em></td>
</tr>
<tr>
<td>Lists of stars 14 <em>LLL LLI</em></td>
</tr>
<tr>
<td>Lists of stars by const. 14 <em>LLL BY</em></td>
</tr>
<tr>
<td>List of stars in Corvus 0 <em>LI I</em></td>
</tr>
<tr>
<td>List of stars in Volans 0 <em>LI I</em></td>
</tr>
<tr>
<td>List of stars in Pavo 0 <em>LI I</em></td>
</tr>
</tbody>
</table>

Figure 2.2: Sample annotations for *Stars.*
of the title for the annotation of taxonomic relations, we relied on information contained in documents linked from the References and External Links sections.

There are cases in which the annotation cannot be consistent with Wikipedia because the information in Wikipedia itself is contradictory. For example, Nemesis (star) is defined in Wikipedia as “a hypothetical hard-to-see red dwarf star or brown dwarf, orbiting the Sun”\(^\text{11}\). With respect to both the root category Stars and the parent category Hypothetical stars, Nemesis seems to be an instance: its title clearly disambiguates to a star and its definition mentions that it can be a hypothetical red dwarf star. However, according to the definition, Nemesis can also be a brown dwarf, which is not a star, since according to Wikipedia “brown dwarfs are sub-stellar objects which are too low in mass to sustain stable hydrogen fusion”\(^\text{12}\). We decided to annotate these conflicting information cases with a special CO tag.

Annotation was sometimes difficult, as some node pairs only partially satisfy the label definitions. Kappa Piscium, for example, is a complex of three stars, of which only the brighter star is specified to be variable. Consequently, we decided to label it as \(O\) with respect to its parent, Variable stars.

In the following section we describe a series of approaches to binary taxonomic relation extraction in which the tag \(O\) is mapped to the negative class of subsumption relations. All the other tags are mapped to the positive class of subsumption relations, covering both class-instance and class-subclass relations. Considering tags other than \(I\) or \(C\) as positive was meant to lead to an increase recall in extracted relations through the use of transitivity. For example, if the two pairs (Films by culture, Films), manually tagged as \(BY\), and (Asian films, Films by culture), manually tagged as \(C\), are classified as positive, then transitivity can be used to also infer that (Asian films, Films) is a positive pair. We leave for future work the use of the same dataset

\(^{11}\) http://en.wikipedia.org/wiki/Nemesis_(star) 
for training and evaluating systems on the task of identifying the finer grained taxonomic labels\textsuperscript{13}.

For each root category in the dataset, Figure 2.1 lists the total number of descendants extracted by the generator algorithm, the number of descendant titles that are annotated as positive with respect to the root category (root-positive), and the number of titles that are annotated as positive with respect to the parent category in the category tree (parent-positive). Across all 10 root categories in the dataset, there are 2,872 titles of which 1,908 are annotated as root-positive, and 2,431 as parent-positive. In order to estimate the inter-annotator agreement (ITA), we selected the Corporations subgraph and asked a second annotator to label the corresponding 260 nodes, using the same annotation guidelines. With respect to the root, the resulting ITA was 96.0%. With respect to the parent, the ITA was 87.5\% - this lower ITA reflects an increased difficulty in tagging parent relations, due to the wide diversity of parent categories in the subgraph. When computed over the original finer grained labels, the ITA was 92.1\% with respect to the root, and 81.6\% with respect to the parent category.

\textsuperscript{13} The dataset will be made available online.
3 TAXONOMIC RELATION EXTRACTION AND EXTRACTION FEATURES

3.1 Taxonomic Relation Extraction

The dataset described in the previous section can be used to train and evaluate systems that extract subsumption relations such as *is-a* and *instance-of* between categories and articles from the Wikipedia category graph. In particular, we have trained and evaluated a series of relation extraction systems that are aimed at two major types of taxonomic relation extraction:

1. **Flat**: Given a category name \( C \), find all descendants of \( C \) in the Wikipedia graph \( WP(C) \) rooted at \( C \) that are true instances or subcategories of \( C \).

2. **Hierarchical**: Given a Wikipedia category graph \( WP(C) \) rooted at a category \( C \), extract a taxonomy subgraph from \( WP(C) \). A graph is a *taxonomy graph* if all edges connect nodes that are in a true subsumption relation.

Flat extractions can be used for building dictionaries of instances and subcategories, which may benefit a wide variety of tasks in natural language processing. In Question Answering, for example, the processing of factual questions such as “What kind of stars ...” or “What stars ...” could be improved through the use of dictionaries that list types of stars, or names of stars. Hierarchical graphs have the added advantage of compactly representing multiple sets of flat extractions. The following sections describe a series of relation classification approaches that were trained and evaluated on the subgraph datasets under the two major scenarios. To simplify the description, we will use \( \{(N,R)\} \) to refer to the subset of annotations that are relative to the root category, and \( \{(N,P)\} \) to denote the subset of annotations that are relative to the parent nodes, in each subgraph dataset. With the
exception of the shared annotations for the first level of nodes under the root category, the two subsets \{((N, R))\} and \{((N, P))\} define a partition of the entire set of annotations.

3.1.1 Flat Taxonomic Relation Extraction

In this setting, the aim is to train a relation extraction system to identify which of the relation pairs in the \{((N, R))\} subset are positive subsumption relations. Depending on the types of relations used during training and the strategy for identifying positive pairs at test time, we defined and evaluated 4 relation extraction systems:

1. Train on \{((N, R))\}, test on \{((N, R))\}. As will be shown in the experiments section, one problem with this simple approach is that it misses too many positive relations.

2. Train on \{((N, R))\}, test on \{((N, R))\} level by level. To alleviate the problem of low recall, when classifying a pair \((N, R)\), this system uses a set of additional features between a node \(N\) and its parent node \(P\) that are defined to also use the label of the pair \((P, R)\). This means that at test time, whenever a pair \((N, R)\) is to be classified, all pairs \((P, R)\) between \(N\)’s parents and the root category need to have been already classified. Given that the Wikipedia category network is a directed acyclic graph, this constraint will be satisfied if the nodes are classified in a topological sort order, using breadth first traversal.

3. Train on \{((N, R))\} \cup \{((N, P))\}, test on \{((N, R))\}. This is similar to the first approach, except that we also use the labeled edges between nodes and their parents during training.

4. Train on \{((N, R))\} \cup \{((N, P))\}, test on \{((N, R))\}, level by level with greedy transitive closure. Whenever a pair of nodes \((P, R)\) is classified as positive, we find all pairs of nodes \((N, P)\) that can be classified as positive, where \(N \in P.C\) is a child of \(P\), and infer by transitivity that \((N, R)\) is positive too.
Table 3.1: Transitive Closure Classification.

<table>
<thead>
<tr>
<th>GreedyClosure($R, F$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[In]: A root category $R$.</td>
</tr>
<tr>
<td>[In]: A learned classification function $f$.</td>
</tr>
<tr>
<td>1. <strong>foreach</strong> $N \in WP(R)$:</td>
</tr>
<tr>
<td>2. set $N.label \leftarrow -1$</td>
</tr>
<tr>
<td>3. set $Q_1 \leftarrow {R}$</td>
</tr>
<tr>
<td>4. <strong>while</strong> $Q_1$ is not empty:</td>
</tr>
<tr>
<td>5. set $P \leftarrow extractFirst(Q_1)$</td>
</tr>
<tr>
<td>6. <strong>ExploreBFT</strong>($P, R, Q_1, f$)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ExploreBFT($P, R, Q_1, f$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. set $Q_2 \leftarrow P.C$</td>
</tr>
<tr>
<td>2. <strong>while</strong> $Q_2$ is not empty:</td>
</tr>
<tr>
<td>3. set $N \leftarrow extractFirst(Q_2)$</td>
</tr>
<tr>
<td>4. if $N.label = -1$:</td>
</tr>
<tr>
<td>5. $Q_2.append(N.C)$</td>
</tr>
<tr>
<td>6. set $N.label \leftarrow f(N, P)$</td>
</tr>
<tr>
<td>7. if $N.label = +1$:</td>
</tr>
<tr>
<td>8. $Q_1.append({N})$</td>
</tr>
</tbody>
</table>

The algorithm for the system F4 is shown in Table 3.1, where the auxiliary procedure `ExploreBFT` ensures that nodes are visited in topological sort order using Breadth-First Traversal (BFT).
3.1.2 Hierarchical Taxonomic Relation Extraction

In this setting, the aim is to train a relation extraction system to identify which of the relation pairs in the \( \{(N, P)\} \) subset are positive subsumption relations. Thus, the trained system takes as input a Wikipedia category graph rooted at a category \( R \) and classifies all edges in the graph, effectively mining a true taxonomy subgraph. We defined and evaluated 2 relation extraction systems:

1. Train on \( \{(N, P)\} \), test on \( \{(N, P)\} \).

2. Train on \( \{(N, R)\} \cup \{(N, P)\} \), test on \( \{(N, P)\} \). This is similar to the first approach, except that we also use the labeled relations between nodes and the root category during training.

3.2 Relation Extraction Features

During training and evaluation, each pair of nodes \((L, H)\) from \( \{(N, R)\} \cup \{(N, P)\} \) is represented as a vector of features. Many features are based on judging the similarity between two words: \( w_l \) extracted from node \( L \) and \( w_h \) extracted from node \( H \). To find whether two words are similar, we look at 4 different sets for each word: the word itself, the words from all its synsets in WordNet, the words from its hypernym synsets, and the words from its hyponym synsets. We than create all possible combinations of sets between the two words and check if their intersections are not empty, which leads to 16 different features. For example, a Boolean feature will encode whether word \( w_l \) is among the hypernyms of word \( w_h \), while another Boolean feature will encode whether \( w_l \) and \( w_h \) have a hypernym in common. In describing the features, we will use the term genus phrase to refer to the phrase in the genus-differentia definition sentence of a Wikipedia article that indicates the genus of the described entity or concept. “A supernova is a stellar explosion that is more energetic than a nova”\(^\text{14}\) is the definition sentence in the Supernova article, consequently “stellar

\(^{14}\) http://en.wikipedia.org/wiki/Supernova
explosion” is extracted as the genus phrase for this article. We also extend the genus phrase attribute to those category pages that are linked to a main article page and equate it with the genus word of the linked article. For example, the Galaxies category is linked to the Galaxy article through the cat main tag, consequently we use “massive, gravitationally bound system” as the genus phrase for the category. We use the term infobox name to refer to the name of the infobox contained directly in the article page or indirectly in the article linked from a category page. We also use the term disambiguation phrase to refer to the phrase between parentheses in some titles, as for example in Rocky (comic strip).

Following is a list of the atomic features used for subsumption classification:

1. $L$ is an article/category page ($H$ can only be a category page).
2. $L/H$ has a proper name title, e.g. the title has multiple content words that are capitalized, as in Artists Open House.
3. $L/H$ is a list/lists node, e.g. its title starts with “list/lists of”, or ends with “list/lists”, as in List of modern artists.
4. $L/H$ is an organizational category, e.g. the title has the pattern “X by Y”, as in Artists by genre.
5. The head word of $L/H$ is in WordNet (useful in combination with other atomic features).
6. The head of $H$ is contained in the head of $L$, as in Viruses and Alphavirus.
7. The head of $H/L$ is a modifier of the head of $L/H$ respectively, as in Beverages and Alcoholic beverage companies.
8. $L$ and $H$ have the same or similar modifiers, but the heads are different.
9. Let $S$ be the set of syntactic heads of the parent categories of $L$ in Wikipedia. Create binary features encoding whether: most of the heads in $S$ are similar with the head of $H$; at least one of the heads in $S$ is similar with the head of $H$. For example, if $L$ is John Hartung, its parent head set $S$ in Wikipedia contains births, people, alumni,
faculty, photographers, artists. For $H = \text{Artists}$, more than half of the heads in $S$ are similar with the head of $H$, i.e. they belong to the synonym/hypernym/hyponym set of the word artist in WordNet.

10. Similarity features between: the head words of $L$ and $H$; the head of the disambiguation phrase of $L/H$ and the head of $H/L$; the head of the genus phrase of $L/H$ and the head of $H/L$; the head of the infobox name of $L/H$ and the head of $H/L$.

11. The tf.idf cosine similarity between the definition sentences of $L$ and $H$.

12. The tf.idf cosine similarity between the article text of the $L$ and $H$.

13. Percentage of overlap between the infobox attributes of $L$ and the extended infobox attributes of $H$.

The infobox attributes of $L$ contain the names of all attributes appearing in an infobox used inside the article $L$ or the main page linked from category $L$. The extended infobox attributes of $H$ refer to the attributes appearing in $H$ or in articles in the dataset that are very likely to be subsumed by $H$, such as articles that share the same head with $H$.

Note that even though some of the similarity features are based on WordNet, the extracted relations do not necessarily have to be consistent with WordNet. The fact that “explosion” and “star” are dissimilar concepts according to WordNet will lead the system to decide that Supernova is not subsumed by Stars, which is the opposite of their WordNet relation.

Since anaphoric noun phrases often indicate the category of the corresponding grounding instance, a separate set of features was designed to exploit coreference links. Given an article title, we use a top scoring subset of the coreference heuristics described in Raghunathan et al. (2010) to identify noun phrases that are coreferent with the title. The syntactic head of each coreferent noun phrase is extracted and used to create the following coreference informed features:
14. At least one of the coreference heads for article $L$ is the same or similar with the head of $H$. For example, $L = \text{Guido Marzulli}$ is coreferent with the predicate nominal \textit{figurative Italian painter}, whose head is a hyponym of the head of $H = \text{Artists}$.

15. The definition sentence of $L$ contains a word that is not coreferent with $L$ and that is the same or similar with the head of $H$. For example $L = \text{Medal theft}$ is defined as “the theft of awards for military action, civil service, and achievements in science or sports”\textsuperscript{15}. The noun \textit{awards} is not coreferent with $L$ and it is the head of $H = \text{Awards}$.

We also include atomic features that encode whether the title of a category $L$ refers to a disjunctive category and is thus unlikely to be a real subcategory of another category $H$:

16. Category title $L$ can be parsed as a conjunction or disjunction of two or more noun phrases, and their syntactic heads are similar, for example \textit{Stained glass artists and manufacturers}.

17. The title $L$ or its disambiguation phrase consists of the head of $H$ followed by words such as \textit{chains, series, serials}, as in \textit{Restaurant chains} or \textit{Abbott and Costello (film series)} with respect to \textit{Restaurant} or \textit{Films}.

More complex features are constructed from conjunctions of two or more atomic features, as for example:

18. $H$ is a list page or an organizational page, conjoined with similarity features between the heads of $L$ and $H$.

19. $L$ is an article, conjoined with similarity features between the heads of $L$ and $H$.

In the flat extraction setting, for any given node $N$ in the dataset the generic features above are computed both between $N$ and the root category $R$ and between $N$ and its direct parents

\textsuperscript{15} http://en.wikipedia.org/wiki/Medal_theft
In systems F2 and F4 that use level by level classification, we add atomic and complex features that use the label calculated for the pair \((N, P)\), such as:

20. \((N, P)\) was classified as positive.

21. \((N, P)\) was classified as negative, and \(P\) is not a list page or an organizational page.
4 SYSTEM IMPLEMENTATION

The taxonomy extraction system is comprised of four modules: the Dataset Generator, the Information Collector, the Feature Extractor and the Relation Classifier (Model Trainer and Classifier make up of this module).

Figure 4.1: The Dataset Generator of the Taxonomic Relation Extraction Systems.
To simplify the explanation of the system implementation, we refer to Figure 4.4 as a sample input (in fact, it is a sample output of the Dataset Generator) to the major part of the system (not including the Dataset Generator) when introducing how each system module (except the Dataset Generator) works and processes the input data.

The Dataset Generator is used to generate a subgraph for a given root category that preserves as much as possible the distribution of the original descendant categories and articles in Wikipedia. It creates the datasets which, after manual annotation, can be used to train and evaluate the relation extraction system.

The Information Collector takes the datasets created by Dataset Generator and performs several functions shown in Figure 4.5:
1. The *Parser* extracts the syntactic head of a given category or article title. In linguistics, the syntactic head of a phrase is the word which controls the syntactic type of this phrase. For example, in the *big red dog*, the word *dog* is the head, as it determines that *big red dog* is a noun phrase and the word *big* and *red* are only modifiers of this head noun. In our systems, most of the time the head words we intend to extract are the same as defined in the linguistics sense. However, there are exceptions. For example, in the *list of artworks by Louise Bourgeois*, the word *list* is the head in the linguistics sense. Meanwhile in our systems, since this page plays its role as an organizational article, in which lists a set of artworks by Louise Bourgeois, the word *list* unleashes few clues to what this Wikipedia article is really about; the word *artworks*, in the other hand, catches the real content of this article, and therefore we choose the word *artworks* to be the head of this article title, as shown in Figure 4.4.
2. For a given syntactic head word, in order to implement similarity features, the *WordNet Information Extractor* collects all of its synset words of the first word sense.
in WordNet, the words from its hypernym synsets which are not beyond the second level and the words from all of its hyponym synsets for the first word sense. As shown in Figure 1.2, Figure 1.3 and Figure 1.4, for the word “actor”, the WordNet Information Extractor collects “actor,” “histrion,” “player,” “thespian” and “role player” as synset words; it chooses “performer,” “performing artist” and “entertainer” as hypernym words, not including “person,” “individual” and other words which are beyond the second level; and it includes “actress,” “leading lady,” “starlet” and all the other hyponyms which are not listed into the hyponym words.
3. In order to implement features related to the disambiguation phrase, for a given category or article name, the Disambiguation Phrase Extractor extracts the disambiguation phrase, if any. The syntactic head word of the disambiguation phrase as well as its related WordNet synset information\textsuperscript{16} are also generated. For example, for the node “Clamp (manga artists)” in Figure 4.4, “manga artists” is extracted as the disambiguation phrase, “artists” is identified as the head word and the WordNet synset information of “artist” is generated by the WoreNet Information Extractor as well.

4. Given a category or article page, the Mapper identifies its true article page in order to identify the infobox name, the definition sentence and coreferent noun phrases. For a category page, the true article page, refers to the main article page for this particular category. Meanwhile, if an article page does not redirect from another page, then it is the true article page. Otherwise, the page which it redirects from is the true article page. For example, as shown in Figure 4.6, the true article page of the category “Artisans” is the article “Artisan.”

5. For a given category or article name, the Infobox Extractor identifies the name of the infobox contained in its true article page to implement features related to the infobox name. An Infobox is used as a summary of some common aspects that are shared by some similar articles; it appears as a table on the top right-hand corner in some Wikipedia articles. As shown in Figure 1.6, the infobox name for “Clamp (manga artists)” is “comics studio.” Similarly, the syntactic head word “studio” is extracted as the head and its related WordNet synset information is also generated.

\textsuperscript{16} For each word in WordNet, the synset information refers to all of its first word sense synset words in WordNet, the words from its hypernym synsets which are not beyond the second level and the words from all of its hyponym synsets of its first word sense.
6. For a given category or article title, the *Redirect Pages Collector* collects all of its redirect pages, if any, in order to implement features related to redirect pages. As shown in Figure 4.7, the article titles “Omniplex Science Museum” and “Oklahoma Science and Arts Foundation” are collected as redirect pages. The syntactic head words of the redirect pages as well as the corresponding *WordNet synset information* are generated at the same time.

![Figure 4.6: The true article page of the category “Artisans.”](image)

7. To implement features which use all the parent categories of a given category or article page, the *Parent Categories Collector* collects all the parent categories in Wikipedia for a given category or article title. As shown in Figure 4.6, the parent categories “Artists,” “Crafts” and “Art occupations” are collected by the *Parent Categories Collector*. The head word of each category as well as its *WordNet synset information* are also generated.
8. Finally, to implement features which are designed to exploit coreference links, the Coreference Extractor collects all the coreferent noun phrases throughout the true article page. As shown in Figure 1.6, the word “group” is collected as one coreference word for the article “Clamp (manga artists).” The syntactic head word of each coreferent noun phrase as well as its corresponding WordNet synset information are also generated.

The Feature Extractor takes the output of the Information Collector as input along with the annotated relations between categories and articles from the newly created Wikipedia category subgraph. Given that all the information required to extract features has already been generated by the Information Collector, the Feature Extractor takes all the required information, generates a feature vector for each relation pair and outputs the feature vector along with the label of the relation pair.

The Relation Classifier takes part of the output of the Feature Extractor to train a taxonomic relation classifier by using LibSVM (Scholkopf and Smola, 2001). Then, given a new relation pair from the testing dataset it runs the Feature Extractor again to generate the feature vector for the relation pair. The trained taxonomic relation classifier then takes the feature vector of the relation pair as input and classifies the relation.

4.1 Dataset Generator

This module implements the function introduced in Chapter 2. Ten diverse categories (shown in Figure 2.1) are selected as root nodes from near the top of the hierarchy for the dataset generator algorithm shown in Table 2.1.

To implement this algorithm, we download the January 16, 2010 Wikipedia xml dump and import it into a MySQL database. By using the Java API JDBC\textsuperscript{18}, the program can

\textsuperscript{17} In the dataset, there are 10 category graphs. Nine out of 10 categories are used to train the system, and the remaining one is used to evaluate the system. This process is repeated 10 times.

\textsuperscript{18} http://dev.mysql.com/downloads/connector/j/
access the MySQL database and therefore generate random paths based on the algorithm shown in Table 2.1.

According to the annotation guidelines shown in Chapter 2, we annotate each relation pair (both node-to-root and node-to-parent) in the 10 subgraphs. At this point, the datasets for both training and evaluation are complete.

4.2 Information Collector

The eight functions implemented by the Information Collector module are introduced in the following subsections.

4.2.1 Parser

This atomic module is used to extract the syntactic head of a phrase based on the following ordered set of rules:

- Step 1: If this phrase has more than one word in it and this phrase begins with “List/Lists of,” or ends with “list/lists,” then take the part after “List/Lists of” or before “list/lists” as the current head word. Repeat this rule until there is no more “List/Lists of” or “list/lists” in it. For example, as shown in Figure 4.4, the head of the phrase “List of artworks by Louise Bourgeois” after this step is “artworks by Louise Bourgeois.” Then the processing of the remaining phrase goes to step 4.

- Step 2: If this phrase has more than one word in it and there is a clause in this phrase, then take the part before the clause sentence as the current head word. Repeat this rule until there is not a clause in it. For example, as shown in Figure 4.4, the head of the phrase “Artists who died in Nazi concentration camps” is “Artists” and the processing of this phrase ends here.

- Step 3: If this phrase has more than one word in it and there is a past participle in this phrase, then take the part before the past participle as the current head word.
Repeat this rule until there are no more past participles in it. For example, as shown in Figure 4.4, the head of the phrase “Films directed by Satyajit Ray” is “Films” and the processing of this phrase ends.

• Step 4: If this phrase has more than one word in it and there is a preposition, such as “by,” “of,” “in,” then take the part before the preposition as the current head word. Repeat this rule until there are no more prepositions in it. For instance, continue from the example of step 1, the head of the phrase “artworks by Louise Bourgeois” is “artworks.” Up to this point, the processing of the phrase “List of artworks by Louise Bourgeois” reaches to the end, and the final head of this phrase is “artworks.”

• Step 5: If the current head word still has more than one word in it, then take the last word in the current head word as the final syntactic head word for this phrase. For example, as shown in Figure 4.4, the head of the phrase “Artist studios” is “studios.”

To implement this parser, we need to use the java API edu.stanford.nlp.tagger\(^{19}\) to identify the past participles in the phrases.

4.2.2 WordNet Information Extractor

To implement the similarity features, we need to generate the *WordNet synset information* for words involved in feature extractions. Table 4.1 shows the algorithm for this module. We use the example in Figure 1.2 to explain the algorithm, the input word \(W = \text{“actor”} \) along with its selected word sense \(I = 1\):

1. In initialization, including line 1 - line 3, set \(S \leftarrow \{\text{actor, histrion, player, thespian, role player}\}, hpe, HPE \leftarrow \{\text{performer, performing artist}\}, hpo, HPO \leftarrow \{\text{actress}\}\.\)

2. From line 4 to line 5, extract the head words for every synset phrase in \(S\) and add them into \(S\): \(S \leftarrow S \cup \{\text{actor, histrion, player, thespian, player}\}\.\)

\(^{19}\) http://nlp.stanford.edu/software/lex-parser.shtml
Table 4.1: WordNet Information Extractor algorithm.

<table>
<thead>
<tr>
<th>WordNet-Information-Generator$(W, I)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[In]: A given word $W$ and its corresponding word sense $I$ in WordNet.</td>
</tr>
<tr>
<td>[Out]: Synset words $S$, hypernym words $HPE$, hyponym words $HPO$.</td>
</tr>
</tbody>
</table>

1. set $S \leftarrow \{\}, HPE \leftarrow \{\}, HPO \leftarrow \{\}$
2. if $W$ exists in WordNet:
   3. set $S \leftarrow W$.synset; $hpe, HPE \leftarrow W$.hyponym hpo, $HPO \leftarrow W$.hyponym
   4. for each $s \in S$:
      5. set $sHead \leftarrow \text{ExtractSyntacticHead}(s), S \leftarrow S \cup sHead$
      6. while $hpe$ does not reach the second level hypernym of $W$:
         7. set $hpe \leftarrow hpe$.hyponym, $HPE \leftarrow HPE \cup hpe$
   8. for each hype $\in HPE$:
      9. set $hypeHead \leftarrow \text{ExtractSyntacticHead}(hype), HPE \leftarrow HPE \cup hypeHead$
   10. while $hpo$ has hyponym:
       11. set $hpo \leftarrow hpo$.hyponym, $HPO \leftarrow HPO \cup hpo$
       12. for each hypo $\in HPO$:
          13. set $hypoHead \leftarrow \text{ExtractSyntacticHead}(hypo), HPO \leftarrow HPO \cup hypoHead$
14. return $S, HPE, HPO$

3. From line 6 to line 7, generate hypernyms of $W$: current $hpe = \{\text{performer, performing artist}\}$, which is the first level hyernym of “actor”; set $hpe \leftarrow hpe$.hyponymPhrases = $\{\text{entertainer}\}$, and add every hypernym phrase in $hpe$ into $HPE$; right now, the $hpe$ reaches the second level hypernym, then it exits the loop.

4. From line 8 to line 9, extract the head words for every hypernym phrase in $HPE$ and add them into $HPE$: $HPE \leftarrow HPE \cup \{\text{performer, artist, entertainer}\}$. 

5. From line 10 to line 11, using depth-first traversal, generate hyponyms of $W$: current $hpo = \{actress\}$, which still has hyponyms; set $hpo \leftarrow hpo.hyponymPhrases = \{leading\ lady\}$, and add every hyponym phrase in $hpo$ into $HPO$. Right now, the $hpo$ does not have any hyponyms, so go back to the previous $hpo = \{actress\}$, which has a second hyponym; set $hpo \leftarrow hpo.hyponymPhrases = \{starlet\}$, and add every hyponym phrase in $hpo$ into $HPO$. Now the $hpo$ does not have any hyponyms, and the previous $hpo = \{actress\}$ does not have a non-visited hyponym either, so it exits the loop.

6. From line 12 to line 13, extract the head words for every hyponym phrase in $HPO$ and add them into $HPO$: $HPO \leftarrow HPO \cup \{actress, lady, starlet\}$.

7. On line 14, return $S = \{actor, histrion, player, thespian, role\ player\}$, $HPE = \{performer, artist\}$, $HPO = \{actress, leading\ lady, starlet, lady\}$.

To implement this algorithm, we need to use the Java API edu.mit.jwi\textsuperscript{20} to collect synsets, hypernyms and hyponyms. The $Parser$ module is also used to extract the syntactic head of each word in synsets, hypernyms and hyponyms.

### 4.2.3 Mapper

In order to extract the infobox name and coreferent noun phrases of a given category or article page, the $Mapper$ maps the given page to its true article page. As shown in Figure 4.6, by searching the pattern “Cat main” through the text, the $Mapper$ identifies the article “Artisans” as the main article page of the category “Artisans.” Then by accessing the Wikipedia database, the $Mapper$ finds the article “Artisans” redirects from the article “Artisan,” which is the true article page.

\textsuperscript{20} http://projects.csail.mit.edu/jwi/
4.2.4 Disambiguation Phrase Extractor

This module integrates the *Parser* and the *WordNet Information Extractor*. Given a category or article title from Wikipedia, the *Disambiguation Phrase Extractor* first extracts the disambiguation phrase between the parenthesis, if any, and then extracts the syntactic head of the disambiguation phrase by the use of the *Parser* as well as generates the related *WordNet synset information* using the *WordNet Information Extractor*. As shown in Figure 4.4, the disambiguation phrase of the article “Clamp (manga artists)” is “manga artists,” and the head of “manga artists” is “artists,” the *WordNet synset information* of which is also generated.

4.2.5 Parent Categories Collector

This module also integrates the function of the *Parser* and the *WordNet Information Extractor*. Given a category or article title from Wikipedia, the *Parent Categories Collector* accesses the Wikipedia database to collect all of its parent categories in Wikipedia. For each parent category, it extracts the syntactic head word and generates the related *WordNet synset information* as well. For example, as shown in Figure 4.6, the parent categories for the category “Artisans” are “Artists,” “Crafts” and “Arts occupations.” The *Parser* extracts the heads “Artists,” “Crafts” and “occupations” and the *WordNet Information Extractor* generates the related *WordNet synset information*.

4.2.6 Redirect Pages Collector

This module is similar to the *Parent Categories Collector*. Instead of collecting all the parent categories of a given category or article page, it collects all the pages redirecting to the given page. Then by using the *Parser* and the *WordNet Information Extractor*, it extracts the syntactic head and the related *WordNet synset information* for each redirect page.
Figure 4.7: The redirect pages of “Science Museum Oklahoma.”

For example, as shown in Figure 4.7, the redirect pages for the article “Science Museum Oklahoma” include “Omniplex Science Museum” and “Oklahoma Science and Arts Foundation.” The head words “Museum” and “Foundation” are extracted along with their related WordNet sysnet information.

4.2.7 Infobox Extractor

This module integrates the function of the Parser, the WordNet Information Extractor and the Mapper. Given a category or article title, we can get the true article page of the given page by using the output of the Mapper. We access the Wikipedia database and generate the text of the given true article page. By searching certain patterns through the text, we can extract the infobox name of the given page. Then by using the Parser and the WordNet Information Extractor, the syntactic head and the related WordNet synset information for each infobox name are generated. For example, as shown in Figure 1.6, by searching the certain pattern “Infobox comics studio”, the Infobox Extractor identifies
the name of the infobox as “comics studio”. The Parser extracts the head “studio” and the related WordNet synset information is generated by the WordNet Information Extractor.

### 4.2.8 Coreference Extractor

Given a category or article title, the Mapper is able to find its true article page and therefore we get the raw text of the true article page. After processing the raw text, we use the Java API stanford-corenlp-v1.0\(^{21}\) to generate all the coreferent noun phrases of the given title. Also, by using the Parser and the WordNet Information Extractor, we are able to extract the syntactic head and the related WordNet synset information of each coreferent noun phrase. For example, as shown in Figure 1.6, one of the coreferent phrases for the article “Clamp (manga artists)” is “group.”

### 4.3 Feature Extractor

Given the output information collected by the Information Collector, the Feature Extractor generates feature vectors for each relation pair.

To explain the function of the module Feature Extractor, we refer to the example in Figure 4.8. According to the feature description in Chapter 3, initialize a feature vector \( F = \{f_1, f_2, ..., f_{21}\} \) and \( f_1 = f_2 = ... = f_{21} = 0 \). The number of the feature is consistent with the feature description in Chapter 3. For example, \( f_1 \) refers to the first feature “L is an article/category page (H can only be a category page).” The following list contains all the features which the relation pair (Clamp (manga artists), Artists) has:

1. “Clamp (manga artists)” is an article page, \( f_1 = 1 \).
2. The head “Clamp” is in WordNet, \( f_5 = 1 \).

\(^{21}\) http://nlp.stanford.edu/software/corenlp.shtml
3. Among all the parent categories’ heads, at least one of them (“artists” in “Female comics artists”) is similar to the head “Artists” of the category “Artists”, $f_9 = 1$.

4. The head “artists” of the disambiguation phrase “manga artists” is similar to the head “Artists” of the category “Artists”, $f_{10} = 1$.

5. The tf.idf cosine similarity between the definition sentences of $L$ and $H$ is 0.02, $f_{11} = 0.02$. 

Figure 4.8: An example for Feature Extractor.
The output for the relation pair (Clamp (manga artists), Artists) is then complete: the label of their relation +1 along with the feature vector representing the relation pair (1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0.02, ..., 0).

![Figure 4.9: Illustration of k-fold cross validation.](image)

For evaluation, we use 10-fold cross validation, as shown in Figure 4.9, which means in each experiment we use nine out of 10 category subgraphs to train the relation extraction system and the remaining one to test the system. For example, in the first experiment, we use the root category “Artists” as the testing data and the other nine categories in Figure 2.1 are used as the training data; the second time, the root category “Awards” is used as the testing data, the other nine root categories are used as the training data. This experiment repeats 10 times, and each time a different category which has not been utilized as testing data is used as the testing data while the other nine are used as the training data. To train the F1 system, each node-to-root relation pair in the training data is taken as input to the Feature Extractor, and for each relation pair, the Feature Extractor outputs a feature vector along with its relation label \((F_i, t_i)\) (\(F_i\) is a feature vector and \(t_i \in \{+1, -1\}\)), as shown for the example relation pair (Clamp (manga artists), Artists).
4.4 Relation Classifier

As shown in Figures 4.2 and 4.3, given the output feature vectors of the training dataset from the Feature Extractor, the relation extraction model is trained by the component Model Trainer of the Relation Classifier by using the Java API LIBSVM\textsuperscript{22}. During testing, the Classifier component of the Relation Classifier takes as input an arbitrary relation pair and the Feature Extractor is used to generate the corresponding feature vector. The relation pair is then classified by applying the trained relation extraction model on the computed feature vector.

4.4.1 Support Vector Machines

A Support Vector Machine (SVM) (Vapnik, 2000) is used for classifying relation pairs. SVMs attempt to find a hyperplane which maximizes the margin separating the classes.

\textsuperscript{22}http://www.csie.ntu.edu.tw/~cjlin/libsvm/

![Figure 4.10: The Margin in the Max-Margin Classifier.](image-url)
Figure 4.10, the filled circles are negative samples, while the empty circles are positive samples. The lines in this figure represent hyperplanes which separate positive from negative samples and the line in the middle is the maximum margin decision boundary. Support vectors are a subset of samples, which are in red circles, and they determine the location of the maximum margin decision boundary. The margin is the distance between the closest sample and the decision boundary. Assume the data $x$ is linearly separable, and $\phi(x)$ is a feature space transformation, then there exists a hyperplane $w$ such that:

$$ (w^T \phi(x) + b)t > 0 \quad (4.1) $$

where $b$ is an offset and $t$ is the label of the data.

For a binary classification problem, given a decision boundary satisfying Equation 4.1, the margin could be computed as:

$$ \text{margin} = \min_n \left[ \frac{t_n(w^T \phi(x_n) + b)}{\|w\|} \right] \quad (4.2) $$

We need to solve the following optimization problem, in order to maximize the margin.

$$ \arg \max_{w,b} \{ \min_n \left[ \frac{t_n(w^T \phi(x_n) + b)}{\|w\|} \right] \} \quad (4.3) $$

It would be very complicated to solve this optimization problem directly. To make it easier, first convert this problem into an equivalent one. We notice that if we rescale $w$ and $b$, then the distances to the hyperplane are unchanged. So, for the closest samples, set

$$ t_n(w^T \phi(x_n) + b) = 1 \quad (4.4) $$

Therefore, all the samples satisfy the following constraint:

$$ (w^T \phi(x_n) + b)t_n \geq 1, \ n = 1, ..., N \quad (4.5) $$
To this point, to find the parameters $w$ and $b$ which maximize the margin is equivalent with solving the following quadratic programming optimization problem:

Minimize

$$J(w, b) = \frac{1}{2}\|w\|^2$$  \hspace{1cm} (4.6)

Subject to

$$(w^T \phi(x_n) + b)t_n \geq 1, \quad n = 1, \ldots, N$$  \hspace{1cm} (4.7)

Lagrange multipliers $\alpha_n \geq 0$ are introduced to solve this optimization problem. By multiplying each constraint in Equation 4.5 with a multiplier, we get the Lagrangian function:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{n=1} \alpha_n \{t_n(w^T \phi(x_n) + b) - 1\}$$  \hspace{1cm} (4.8)

where $\alpha = (\alpha_1, \ldots, \alpha_N)^T$.

It can be shown that the solution to the optimization problem is as follows (Scholkopf and Smola, 2001):

$$w = \sum_{n=1} \alpha_n t_n \phi(x_n)$$  \hspace{1cm} (4.9)

$$b = \frac{1}{\|S\|} \sum_{n \in S} (t_n - \sum_{m \in S} \alpha_m t_m k(x_n, x_m))$$

where $k(x_n, x_m) = \phi^T(x_n)\phi(x_m)$ is the kernel function and $S$ is the set of support vectors.

The support vectors are the samples for which $\alpha_n$ are not zero. The kernel trick maps samples from a general set into an inner product space, without computing the mapping explicitly (Aizerman et al., 1964). The kernel transformation enables the mapping from the original feature space to a higher dimensional space.
To classify linearly inseparable data, slack variables are introduced to allow maximizing the margin with some of the samples not within the decision boundary. The new constraints become:

\[ t_n(w^T \phi(x_n) + b) \geq 1 - \xi_n, \quad n = 1, \ldots, N \]

where \( \xi_n \) are the slack variables.

The optimization problem becomes:

Minimize

\[ J(w, b) = \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n \]  \hspace{1cm} (4.11)

Subject to

\[ t_n(w^T \phi(x_n) + b) \geq 1 - \xi_n, \quad n = 1, \ldots, N \]  \hspace{1cm} (4.12)

\[ \xi_n \geq 0 \]  \hspace{1cm} (4.13)

where \( C \) is the regularization constant.

A larger \( C \) means more effort is put into reducing training errors, while a smaller \( C \) means minimizing the norm of the weight vector is given more emphasis.

The reasons for using a Support Vector Machine as the classification model are:

1. SVMs obtain the best performance on a set of classification tasks, which range from text classification (Joachims, 2002) to genomic data processing (Pavlidis et al., 2001).

2. The use of kernel functions enables the SVM model to implicitly work in an enriched, high dimensional feature space. For example, with a quadratic kernel, some atomic
features are combined to generate implicit features which are very useful. “Barnard Natan” is an article under the root category “Artists”. When classifying the relation pair (“Barnard Natan”, “Artists”), this relation pair has the atomic feature “$L$ (here “Barnard Natan” is $L$) is an article page”, as well as the atomic feature “The head word of $L$ (here “Natan” is the head word) is not in WordNet”. By implicitly combining these two atomic features, we get a new feature “for a relation pair $(L, H)$, $L$ is an article page, and the head word of $L$ is not in WordNet”. This new feature for the relation pair (“Barnard Natan”, “Artists”) is an indication that “Barnard Natan” is a named entity and the relation pair (“Barnard Natan”, “Artists”) is a positive pair.
5 EXPERIMENTAL RESULTS

Six systems are trained and evaluated on the newly created datasets. As shown in Chapter 3, two major types of taxonomic relations are proposed: flat relation extraction systems and hierarchical relation extraction systems. According to the types of the relations used at training time and the algorithms used to classify relation pairs during testing, we propose and evaluate four flat relation extraction systems and two hierarchical relation extraction systems:

1. F1: Train on \{(N, R)\}, test on \{(N, R)\}. This simple approach misses too many positive relations, which will be shown in the experiment results.

2. F2: Train on \{(N, R)\}, test on \{(N, R)\} level by level. Considering the problem of the first approach, this approach tries to increase the system recall. In order to do so, this system uses a set of additional features between a node \(N\) and its parent node \(P\), which will be used in the pair \((P, R)\). This set of features require that at test time, whenever a pair \((N, R)\) is to be classified, all pairs \((P, R)\) between \(N\)'s parents and the root category need to have been already classified. The fact that the Wikipedia category graph is a directed acyclic graph could satisfy the constraint of classifying all the parent nodes of the child node before classifying the child node itself, if breadth first traversal is used to classify the nodes in a topological sort order.

3. F3: Train on \{(N, R)\} \cup \{(N, P)\}, test on \{(N, R)\}. This approach is similar to the first one, with the exception that the labeled edges between nodes and their parents are also used during training time.

4. F4: Train on \{(N, R)\} \cup \{(N, P)\}, test on \{(N, R)\}, level by level with greedy transitive closure. Whenever a pair of nodes \((P, R)\) is classified as positive, for each \(N \in P.C\),
which means \( N \) is a child of \( P \), if \((N, P)\) is classified as positive, by transitivity, \((N, R)\) is inferred to be positive too.

5. H1: Train on \{\((N, R)\)\}, test on \{\((N, P)\)\}.

6. H2: Train on \{\((N, R)\) \(\cup\) \((N, P)\)\}, test on \{\((N, R)\)\}. This approach is similar to the first one, with the exception that the labeled relations between nodes and the root category are also used during training time.

We trained and evaluated the flat and hierarchical extraction approaches using 10-fold cross validation, in which 9 category subgraphs are used for training and the remaining 1 category subgraph is used for testing. Each category subgraph was used for testing and the test results were pooled across all 10 subgraphs. For each system we used the LibSVM implementation of Support Vector Machines (SVM) (Scholkopf and Smola, 2001) with a quadratic kernel (Vapnik, 2000).

![Figure 5.1: ROC graph for flat taxonomic relation extraction.](http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
Figure 5.2: ROC graph for hierarchical taxonomic relation extraction.

Table 5.1: Extraction performance.

<table>
<thead>
<tr>
<th>System</th>
<th>Acc</th>
<th>P</th>
<th>R</th>
<th>F₁</th>
<th>ROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>84.6</td>
<td>89.8</td>
<td>86.3</td>
<td>87.1</td>
<td>91.3</td>
</tr>
<tr>
<td>F2</td>
<td><strong>85.2</strong></td>
<td><strong>89.5</strong></td>
<td><strong>88.2</strong></td>
<td><strong>88.8</strong></td>
<td><strong>91.7</strong></td>
</tr>
<tr>
<td>F3</td>
<td>82.7</td>
<td>83.7</td>
<td>91.8</td>
<td>87.6</td>
<td>90.2</td>
</tr>
<tr>
<td>F4</td>
<td>83.0</td>
<td>86.0</td>
<td>88.8</td>
<td>87.4</td>
<td>87.5</td>
</tr>
<tr>
<td>H1</td>
<td><strong>88.7</strong></td>
<td><strong>91.9</strong></td>
<td><strong>95.0</strong></td>
<td><strong>93.4</strong></td>
<td><strong>88.8</strong></td>
</tr>
<tr>
<td>H2</td>
<td>88.0</td>
<td>93.9</td>
<td>91.9</td>
<td>92.9</td>
<td>88.7</td>
</tr>
</tbody>
</table>

Table 5.1 reports the accuracy, precision, recall and F₁ measure for each of the proposed methods. They are computed based on the following measures:

1. True positive (TP): the prediction outcome is positive, and the actual value is also positive.
2. False positive (FP): the outcome is positive, and the actual value is negative.

3. True negative (TN): the outcome is negative, and the actual value is negative.

4. False negative (FN): the outcome is negative, and the actual value is positive.

5. True positive rate (TPR) (recall): $TPR = TP / (TP + FN)$.

6. False positive rate (FPR): $FPR = FP / (FP + TN)$.

7. Positive predictive value (PPV) (precision): $PPV = TP / (TP + FP)$.

8. F1 score: $F1 = 2 \times TPR \times PPV / (TPR + PPV)$.

Furthermore, we use the margin value computed by the SVM as a classification confidence measure to rank the examples during testing and plot the corresponding Receiver Operating Characteristic (ROC) curves, as shown in Figure 5.1 and Figure 5.2. The area under the ROC curve is reported in the last column of Table 5.1. The ROC curves plot TPR vs. FPR and are computed by pooling together and ranking the test examples from all 10 experiments (one experiment per fold). Since the margin values are not directly comparable between examples from different test folds, the comparison between the ROC values of different approaches should be taken with a grain of salt.

The results in Table 5.1 show limited improvement in performance when using the greedy level-by-level classification in the flat extraction approach F2. Combining node-to-parent and node-to-root relations during training hurts performance for both flat and hierarchical extraction, which suggests that there are significant differences between the distributions of these two types of relations. A simple baseline that assigns the most common label to all test examples (the positive label) obtains 66.4% accuracy in the flat setting and 84.6% accuracy in the hierarchical setting. Hierarchical extraction is therefore more difficult, as the absolute improvement over the most common label baseline is around
4%, equivalent with a 26% relative error reduction. We believe that the difficulty in learning a more accurate classifier in the hierarchical setting is caused by the higher diversity of the node-to-parent relations and the unbalanced training data which is heavily skewed towards positive pairs.
6 CONCLUSION AND FUTURE WORK

6.1 Future Work

There are two major directions for future work. The first one is to design a method that automatically updates the newly created datasets in order to make them consistent with later Wikipedia versions. The second direction is to improve the relation extraction systems. We could adjust the implementation to make the extracted head word more accurate and make the selection of word sense for a given word more precise. Future work will also explore more aspects of Wikipedia and integrate corpus-based similarity measures into the system, in hope of improving the performance of the relation extraction.

6.1.1 Updating the Datasets

To make the annotated datasets evolve as Wikipedia changes, one possible solution is to obtain access to the Wikipedia OAI-PMH live feed, which reports instant Wikipedia changes. By getting access to the instant reports about all Wikipedia changes, we could know the pages in Wikipedia that have been modified and that may lead to a modification of our annotations.

6.1.2 Improving the Implementation

Error analysis showed that one cause for mis-classifying relations was the incorrect extracted head word of a given category or article title. To solve this problem, instead of only using a rule-based parser, we could integrate a dependency parser into the original rule-based parser, considering the goal is to extract the head word of a noun phrase.

Another cause for mis-classifying relations is incorrect WordNet Synset information. For the current systems, we choose the most frequently used sense as the word sense of a given word. In future work, we could do word sense disambiguation right before selecting the word sense for a given word.
6.1.3 Corpus Based Similarity Measures

In order to make taxonomic relation extraction more Wikipedia-driven, we plan to use corpus based similarity measures such as Latent Semantic Analysis (Landauer et al., 1998) and Explicit Semantic Analysis (Gabrilovich and Markovitch, 2007).

Latent Semantic Analysis (LSA) is a measure which aims to find out the relations between documents and terms by mapping the documents and the words they contain to a set of concepts. The major assumption of LSA that words which have close meanings will occur close together in text leads to the calculation of the similarity between words. The LSA technique represents the text as a matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) and mathematical techniques are used to reduce the number of columns while preserving the similarity structure among rows. The semantic relatedness between two words is expressed by taking the cosine of any two rows, and the resulting values close to one indicate very similar words while values close to zero indicate very dissimilar words (Landauer et al., 1998).

Explicit Semantic Analysis (ESA) is a method to analyze the semantic relatedness between natural language texts. This method first represents each Wikipedia concept as a weighted vector of words contained in the corresponding Wikipedia article. Then by using inverted index, each word is mapped into a list of weighted Wikipedia concepts. After the previous steps, this method represents texts by a sequence of concepts from Wikipedia. These concepts are weighted according to their relevance to the input text. The task of computing semantic relatedness between texts equals to comparing the vectors representing the texts (Gabrilovich and Markovitch, 2007).

By using LSA and ESA, we could compute the similarity between words, which could be used to replace or add to WordNet when generating feature vectors for a relation pair, and may help improve the relation extraction performance.
6.1.4 Multilingual Information in Wikipedia

We also plan to exploit the multilingual information in Wikipedia by creating features using multilingual information. One possible feature could be: for each node $N_e$ under a root category in the English version, find all cross-linked multi-lingual nodes $N_o$ for this node. For each cross-linked multi-lingual node $N_o$ in languages other than English, find all the super-categories $P_o$ for $N_o$, and put them in the set $S_o$. For each $P_o$ belonging to $S_o$, map it back to the English version $P_e$, and put it in the set $S_e$. After completing the set $S_e$, find the head word for each title in $S_e$. Then check if the head of the root category, which $N_e$ is under, is similar to the head word with highest frequency of the phrases in $S_e$. For example, “Award winners” is a category node under the root category “Awards”. “Prismodtagere” is the corresponding node of “Award winners” in Dannish Wikipedia. “Personer” is one of the super-categories for “Prismodtagere” in Dannish Wikipedia. “People” is the corresponding category in English Wikipedia of “Personer”. Similarly, we can generate $S_e$ for the node “Award winners” and it contains “Awards”, “People”, “People by status”, “People by association” and so on. After extracting the head words of all the titles in $S_e$, we find out that the head word with the highest frequency is “people”. The root category “Awards” is not similar to “people”, which is a strong indication that “Award winners” and “Awards” are not in an is-a relation. We could also apply all the features designed for the English version of Wikipedia to other language versions by the use of natural language processing tools for other languages. For example, for a relation pair $(L, H)$, we find out their corresponding nodes in Chinese $L_c$ and $H_c$. By using NLP tools for Chinese, we extract the head words $W_l$ and $H_l$ of $L_c$ and $H_c$. “$W_l$ is similar to $H_l$” could be a new feature.

6.2 Conclusion

In this thesis, we propose a machine learning method to extract taxonomic relations from Wikipedia. The contributions of this thesis include:
1. We generate a new dataset of category subgraphs which are automatically sampled from the Wikipedia category graph, in which the node-to-parent and node-to-root pairs are manually annotated with *is-a* and *instance-of* relations. These new datasets can be used both for training and evaluating taxonomic relation extraction systems.

2. A set of learning approaches are designed for two major types of taxonomic relation extraction: *flat* extraction of node-to-root relations and *hierarchical* extraction of node-to-parent relations. Trained on the newly created datasets, the taxonomic relation extraction systems are able to take a Wikipedia category graph as input, exploit the structure of Wikipedia through a rich set of features and determine which relation pairs are positive subsumption relations.

3. By using the newly created datasets, we enable a better evaluation of taxonomic relation extraction systems, not only estimating the precision of the system but also obtaining a clear sense of the system’s recall.

For future work, we plan to design a method that automatically updates the datasets in order to make them consistent with later versions of Wikipedia. To compute word-to-word similarities, we intend to use corpus-based similarity measures such as Latent Semantic Analysis and Explicit Semantic Analysis. We also plan to use multilingual information in Wikipedia to generate new features for relation pairs, in order to further increase the relation extraction performance.


APPENDIX: SAMPLE DATASETS

This appendix contains 10 sections, one section for each of the 10 root categories. Each section contains a portion of a dataset, along with a description of that dataset. In each subgraph, the first line is the title of the root category and there are four components for each descendant category or article under this root category. The first part is the title of this category or article node and the second part is the namespace of this node, “0” denoting an article node, “14” denoting a category node. The third part is the label of the relation between this node and the root category, while the fourth part is the label of the relation between this node and its parent.

A.1 Sample Dataset for the Category “Artists”

The annotation labels follow the annotation guidelines described in Chapter 2. In this subgraph, Figure A.1 shows part of the root category “Artists” which has 215 descendant categories and articles.

A.2 Sample Dataset for the Category “Awards”

The annotation labels follow the annotation guidelines described in Chapter 2. In this subgraph, Figure A.2 shows part of the root category “Awards” which has 220 descendant categories and articles.

A.3 Sample Dataset for the Category “Beverages”

The annotation labels follow the annotation guidelines described in Chapter 2. In this subgraph, Figure A.3 shows part of the root category “Beverages” which has 416 descendant categories and articles.
### Artists

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artist studios</td>
<td>O O</td>
</tr>
<tr>
<td>Manga studios</td>
<td>O I</td>
</tr>
<tr>
<td>Clamp (manga artists)</td>
<td>O I</td>
</tr>
<tr>
<td>Riker Hill Art Park</td>
<td>O I</td>
</tr>
<tr>
<td>Artists Open House</td>
<td>O I</td>
</tr>
<tr>
<td>Frank Lloyd Wright Home and Studio</td>
<td>O I</td>
</tr>
<tr>
<td>Hannah Morris</td>
<td>I I</td>
</tr>
</tbody>
</table>

### Murdered artists

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artists who died in Nazi concentration camps</td>
<td>C C</td>
</tr>
<tr>
<td>Actors who died in Nazi concentration camps</td>
<td>C C</td>
</tr>
<tr>
<td>Bernard Natan</td>
<td>I I</td>
</tr>
<tr>
<td>Samuel Jessurun de Mesquita</td>
<td>I I</td>
</tr>
<tr>
<td>Friedl Dicker-Brandeis</td>
<td>I I</td>
</tr>
<tr>
<td>Photographers who died in Nazi concentration camps</td>
<td>C C</td>
</tr>
<tr>
<td>Erich Salomon</td>
<td>I I</td>
</tr>
<tr>
<td>Painters who died in Nazi concentration camps</td>
<td>C C</td>
</tr>
<tr>
<td>Stefan Filipkiewicz</td>
<td>I I</td>
</tr>
<tr>
<td>Masumi Hayashi (photographer)</td>
<td>I I</td>
</tr>
<tr>
<td>Polidoro da Caravaggio</td>
<td>I I</td>
</tr>
</tbody>
</table>

#### Figure A.1: A Portion of the Dataset Artists.

### A.4 Sample Dataset for the Category “Corporations”

The annotation labels follow the annotation guidelines described in Chapter 2. In this subgraph, Figure A.4 shows part of the root category “Corporations” which has 259 descendant categories and articles.
### Awards

<table>
<thead>
<tr>
<th>Category</th>
<th>Level</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Award winners</td>
<td>14</td>
<td>O</td>
</tr>
<tr>
<td>Recipients of the Golden Rose</td>
<td>14</td>
<td>O</td>
</tr>
<tr>
<td>Holmenkollen medalists</td>
<td>14</td>
<td>O</td>
</tr>
<tr>
<td>Lambda Literary Award winners</td>
<td>14</td>
<td>O</td>
</tr>
<tr>
<td>Lambda Literary Award winning books</td>
<td>14</td>
<td>O</td>
</tr>
<tr>
<td>Award ceremony</td>
<td>0</td>
<td>O</td>
</tr>
<tr>
<td>Medal theft</td>
<td>0</td>
<td>O</td>
</tr>
<tr>
<td>Awards by awarding entity</td>
<td>14</td>
<td>BY</td>
</tr>
<tr>
<td>United Nations awards</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>UNESCO awards</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>United Nations Public Service Awards</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>United Nations Medal</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Heinz Award</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Honors of the Holy See</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>Orders, decorations, and medals of the Holy See</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>Orders of knighthood of the Papacy</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>Order of St. Gregory the Great</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>Order of St. Gregory the Great</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Order of Pius IX</td>
<td>0</td>
<td>C</td>
</tr>
</tbody>
</table>

Figure A.2: A Portion of the Dataset Awards.

### A.5 Sample Dataset for the Category “Films”

The annotation labels follow the annotation guidelines described in Chapter 2. In this subgraph, Figure A.5 shows part of the root category “Films”, which has 442 descendant categories and articles.
A.6 Sample Dataset for the Category “Natural disasters”

The annotation labels follow the annotation guidelines described in Chapter 2. In this subgraph, Figure A.6 shows part of the root category “Natural disasters” which has 259 descendant categories and articles.
Corporations

Corporation-related lists 14

Asset lists 14

- List of assets owned by Viacom 0
- List of assets owned by General Electric 0
- List of assets owned by Village Voice Media 0
- List of assets owned by Berkshire Hathaway 0
- List of assets owned by Comcast 0
- List of assets owned by Rogers Communications 0
- List of assets owned by Bertelsmann 0
- List of assets owned by The New York Times Company 0
- List of assets owned by Siemens 0
- List of assets owned by Dow Jones 0
- List of assets owned by Disney 0
- Lists of corporate assets 0
- List of assets owned by CBS 0
- List of assets owned by Wendy’s International, Inc. 0
- List of assets owned by Cablevision 0
- List of assets owned by Hearst Corporation 0
- List of assets owned by Canadian Broadcasting Corporation 0

Figure A.4: A Portion of the Dataset Corporations.

A.7 Sample Dataset for the Category “Restaurants”

The annotation labels follow the annotation guidelines described in Chapter 2. In this subgraph, Figure A.7 shows part of the root category “Restaurants” which has 274 descendant categories and articles.
A.8 Sample Dataset for the Category “Sports clubs”

The annotation labels follow the annotation guidelines described in Chapter 2. In this subgraph, Figure A.8 shows part of the root category “Sports clubs” which has 338 descendant categories and articles.
A.9 Sample Dataset for the Category “Stars”

The annotation labels follow the annotation guidelines described in Chapter 2. In this subgraph, Figure A.9 shows part of the root category “Stars” which has 189 descendant categories and articles.
A.10 Sample Dataset for the Category “Viruses”

The annotation labels follow the annotation guidelines described in Chapter 2. In this subgraph, Figure A.10 shows part of the root category “Viruses” which has 251 descendant categories and articles.
## Sports clubs

<table>
<thead>
<tr>
<th>Squash venues</th>
<th>Toronto Lawn Tennis Club</th>
<th>Racquet Club of Philadelphia</th>
<th>Queen’s Club</th>
<th>Racquet and Tennis Club</th>
<th>Yale Club of New York City</th>
<th>Tuxedo Club</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

## Australian rules football clubs

<table>
<thead>
<tr>
<th>AFL Canberra clubs</th>
<th>AFL Canberra</th>
<th>Manuka Football Club</th>
<th>Eastlake Football Club</th>
<th>Bordeaux Bombers</th>
<th>Paris Cockerels</th>
<th>Strasbourg Kangaroos</th>
<th>East Fremantle Football Club</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure A.8: A Portion of the Dataset Sports clubs.
Stars

Stars with proper names 14 C C

Delta Tauri 0 I I
Delta Geminorum 0 I I

Traditional Star Names 0 C C

Lists of stars 14 LLI LLI

Lists of star names by constellation 14 BY BY

List of star names in Lepus 0 LI LI
List of star names in Pisces 0 LI LI
List of star names in Crux 0 LI LI

List of brown dwarfs 0 LI LI

Navigational stars 0 C C

Lists of stars by constellation 14 LLI LLI

Lists of star names by constellation 14 LLI LLI

List of stars in Corvus 0 LI LI
List of stars in Volans 0 LI LI
List of stars in Pavo 0 LI LI

Deneb el Okab 0 SI SI
Keun Nan Mun 0 SI SI

List of extrasolar planet extremes 0 O O

Figure A.9: A Portion of the Dataset Stars.
<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bunyaviruses</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>Phleboviruses</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Nairovirus</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Tospovirus</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Togaviruses</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>Togaviridae</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Ross River virus</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Alphavirus</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Sindbis virus</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Venezuelan equine encephalitis virus</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Babanki virus</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Picornavirales</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>Nepovirus</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Influenza</td>
<td>14</td>
<td>O</td>
</tr>
<tr>
<td>National Influenza Centres</td>
<td>14</td>
<td>O</td>
</tr>
<tr>
<td>National Influenza Centers</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Influenza vaccines</td>
<td>14</td>
<td>O</td>
</tr>
<tr>
<td>Optaflu</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>Influenza A virus subtype H5N1</td>
<td>14</td>
<td>C</td>
</tr>
</tbody>
</table>

Figure A.10: A Portion of the Dataset Viruses.