

**STEP: SET OF T-UPLES EXPANSION
USING THE WEB**

LIU YUGANG
(B.Comp(Hons), Shandong University)

**A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF
SCIENCE**

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2011

Acknowledgements

I have really appreciated my supervisor, friends and family for all the help and support during my work on this thesis.

I would give my sincere thanks to my supervisor, Prof. Bressan Stéphane. Without his sensitive clairvoyance and inspiration for research, the STEP idea can never be born. During numerous discussions with him, I gradually realize how to work creatively and productively. Moreover, I learn a lot of experience and truth from him, especially to way to live with enthusiasm and optimism.

I am deeply grateful to Dr. Bajleet Malhotra for his great assistance. All the valuable suggestions throughout my thesis work deserve my sincere thanks. I would also thank his family who understand and support his cooperation with me. I would like to wish you and your family wellness and happiness.

I am also grateful to Dr. Panagiotis Karras for his comments and suggestions earlier in my thesis writing, which defenses me and my work in a safe position.

My special thanks are given to Prof. Tan Tiow Seng who gives me the valuable opportunity to study here, and also encourages me a lot. It is him who gave me the support to go through a tough time in my studying here.

The final gratitude is dedicated to my parents and my brother for all their love and support they give me so far. They are the source of impetus and spiritual pillar from which I have drawn power and energy for coping with challenges and accomplishing this thesis. I love you.

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Set Expansion	3
1.3	Contributions	8
1.4	Plan	9
2	Related Work	10
2.1	Taxonomy of Set Expansion Related Techniques	10
2.1.1	Taxonomy Based on Data Source	11
2.1.2	Taxonomy Based on Pattern Construction	12
2.1.3	Taxonomy Based on Arity of Seeds and Target Relations	13
2.2	Representative Work	14
2.3	Comparison	16
3	Background	19
3.1	DIPRE	19
3.1.1	Step One: Fetch Relevant Documents	20
3.1.2	Step Two: Construct Patterns and Extract Candidates	21
3.1.3	Step Three: Rank Candidates	24
3.1.4	Performance Evaluation	24
3.2	SEAL	25
3.2.1	Step One: Fetch Relevant Documents	26
3.2.2	Step Two: Construct Patterns and Extract Candidates	27
3.2.3	Step Three: Rank Candidates	30
3.2.4	Performance Evaluation	31
3.2.5	Extend SEAL for Binary Relation Extraction	32

4	STEP: Set of T-uples Expansion	34
4.1	Problem Formulation	35
4.2	Overview of STEP	36
4.2.1	Step One: Fetch Relevant Documents	37
4.2.2	Step Two: Construct Patterns and Extract Candidates	38
4.2.3	Step Three: Rank Candidates	39
4.3	Step Two: Construct Wrappers and Extract Candidates	40
4.3.1	Regular Expression Based Wrappers	40
4.3.2	Extracting T-uples from Sibling Pages	45
4.4	Step Three: Rank Candidates	51
4.5	Bootstrapping of STEP	55
5	Performance Evaluation	58
5.1	Datasets	58
5.2	Evaluation Metric	61
5.3	Results	62
5.4	Discussions	74
6	Conclusion and Future Work	76
6.1	Conclusion	76
6.2	Future Work	78
	Bibliography	79
A	Datasets Description and Results Illustration	84
A.1	D1	84
A.2	D2	85
A.3	D3	86
A.4	D4	88
A.5	D5	89

A.6 D6	89
A.7 D7	90
A.8 D8	91
A.9 D9	93
A.10 D10	94
A.11 D11	95
A.12 D12	96
A.13 D13	97
A.14 D14	98
A.15 D15	99

Summary

Set expansion is the task of finding members of a semantic class, the set, given a small subset of its members, the seeds. Set expansion systems have leveraged the explosion of the number of HTML formatted lists of all sorts and kinds on the World Wide Web. Such syntactical set expansion from the Web works particularly well for the expansion of sets of atomic values. In this thesis, we present STEP, a set of t-uples expansion system. STEP extends the SEAL set expansion system [Wang 2007] to the expansion of set of t-uples, or relations as in Codd's relational model. The generalization from sets of atomic values expansion to set of t-uples expansion raises problems at every stage of the expansion process, mainly, location of the sources, wrapper (specific contexts that bracket the seeds) construction and extraction of candidates, and ranking of candidates. We therefore argue that set of t-uples expansion compels extensions to the existing expansion process as proposed by many solutions including SEAL. We show that set of t-uples expansion can be achieved effectively by: (i) making the wrappers more flexible, (ii) expanding the search to more pages, in particular to the collections of pages that belong to a same website as t-uples may be located on multiple pages rather than on a same page, and (iii) considering more entities, such as domains, to improve the ranking of candidates. We empirically evaluate the performance of STEP. We compare the successive techniques that we introduce with the baselines provided by SEAL and show significant improvement. Besides, we also study different factors that can affect the performance of STEP and offer some constructive suggestions.

List of Tables

3.1	Five seed books used in DIPRE [Brin 1998].	20
3.2	Example of an occurrence in DIPRE.	22
3.3	Experimental statistics of DIPRE.	25
3.4	HTML codes for a Web page.	29
3.5	One wrapper and two candidates on the Web page in Table 3.4.	29
3.6	Nodes and relations in the graph in SEAL (from [Wang 2007]).	30
3.7	Explanation for each dataset (* are incomplete sets) (from [Wang 2007]).	31
3.8	Five datasets for evaluating relational SEAL (adapted from [Wang 2009]).	33
4.1	Top five URLs of query 1 returned by Google.	37
4.2	Top five URLs of query 2 returned by Google.	37
4.3	Demonstration of wrapper construction on a Web page.	43
4.4	An example of wrapper	45
4.5	Two sibling pages from "marinetraffic.com".	46
4.6	Parameters description.	50
4.7	Procedures used in the Procedure FetchSeedPages, ExtractOverSib- lingPages, and BuildGraph.	50
4.8	The nodes and their relations in the graph.	52
4.9	Top ten candidate t-uples after one iteration.	56
5.1	Baseline datasets used in the performance evaluation.	59
5.2	Parameter setting.	62
5.3	Comparison of accuracy of DIPRE and STEP with varying size of randomly choosing set ($ \theta = 20, 30, 50, 100$).	63

5.4	Comparison of precision of top N_c ($N_c = 10, 20, 50, 100$) candidates returned by SEAL and STEP).	64
5.5	Comparison of recall of top N_c ($N_c = 10, 20, 50, 100$) candidates returned by SEAL and STEP).	64
5.6	Comparison of precision and recall of top 20 candidates with varying number of seeds ($N_s = 2, 4, 6, 8, 10$).	66
5.7	Comparison of precision and recall of top 20 candidates with varying arity of seeds and target relations ($N = 2, 3, 4$).	66
5.8	Comparison of precision of top N_c ($N_c = 10, 20, 50, 100, 200$) candidates with and without extraction over sibling pages.	67
5.9	Comparison of recall of top N_c ($N_c = 10, 20, 50, 100, 200$) candidates with and without extraction over sibling pages.	67
5.10	Comparison of domain ranking of STEP and Google Toolbar on D7.	68
5.11	Comparison of precision of top 100 candidates with varying number of Web pages ($N_p = 10, 20, 50, 100$).	69
5.12	Comparison of recall of top 100 candidates with varying number of Web pages ($N_p = 10, 20, 50, 100$).	69
5.13	Comparison of precision of top N_c ($N_c=10, 20, 50, 100$) candidates with different choices of seeds.	70
5.14	Another example of wrapper	70
5.15	Top ten Web pages ranked by PageRank.	73
5.16	Top ten Web pages ranked by frequency.	74
A.1	Parameter setting of STEP.	84

List of Figures

1.1	Snapshot of Boo!Wa!	3
1.2	Output of Boo!Wa!	4
1.3	Snapshot of Google Sets.	5
1.4	Output of Google Sets.	7
1.5	A three-step framework of set expansion systems.	8
2.1	A taxonomy of set expansion related systems.	17
3.1	Duality between patterns and relations.	20
3.2	Flow chart of SEAL (from [Wang 2007]).	26
3.3	Top URLs containing " <i>Ford</i> ", " <i>Toyota</i> " and " <i>Nissan</i> " returned by Google.	27
3.4	Pseudo-code for wrapper construction of SEAL (from [Wang 2009]).	28
4.1	Architecture of STEP.	36
4.2	Snapshot of a Web page containing amateur radio magazines.	44
4.3	Schema for extracting t-uples from sibling pages.	47
4.4	Example of part of an entity graph.	55
5.1	Comparison of precision of top 20 candidates in different iterations ($i = 1, 2, 3, 4, 5$).	71
5.2	Comparison of recall of top 20 candidates in different iterations ($i = 1, 2, 3, 4, 5$).	72

List of Algorithms

1	DIPRE's algorithm	21
2	GenerateOnePattern(O) (adapted from [Brin 1998]).	22
3	GeneratePatterns(O) (adapted from [Brin 1998]).	24
4	FindOccurrenceOnOnePage(S, d).	41
5	GenerateWrappers(S, d).	42
-	Procedure FetchSeedPages($N_p, Seeds$)	47
6	FindOccurrenceOnSiblingPages(S, D).	48
7	GenerateWrappersOverSiblingPages(S, D).	49
-	Procedure ExtractOverSiblingPages($N_p, N, Seeds$)	49
-	Procedure BuildGraph($N_p, N, Seeds$)	53
8	ExtractOverSiblingPages'($N_p, N, Seeds$)	54
9	Bootstrapping algorithm of STEP	56

List of Acronyms

DIPRE	Dual Iterative Pattern Relation Expansion
DS	Distributional Similarity
IE	Information Extraction
IMO	International Maritime Organization
IR	Information Retrieval
MRR	Mean Reciprocal Rank
NER	Named Entity Recognition
NLP	Natural Language Processing
PMI	Pointwise Mutual Information
POS	Part-Of-Speech
PU Learning	Positive and Unlabeled examples Learning
SAC	Schema Auto Completion
SEAL	Set Expander for Any Language
STEP	Set of T-uples ExPansion using the Web
TF-IDF	Term Frequency - Inverse Document Frequency
URL	Uniform Resource Locator
WI	Wrapper Induction
WSD	Word Sense Disambiguation
WWW	World Wide Web

List of Symbols

I	Number of iterations in a bootstrapping process
N	Arity of seeds and candidate t-uples
N_c	Number of top candidate t-uples
N_p	Number of Web pages returned by a search engine
N_s	Number of seed t-uples
<i>siblingPage</i>	A boolean flag indicating whether extracting t-uples from sibling pages

Introduction

Contents

1.1	Motivation	1
1.2	Set Expansion	3
1.3	Contributions	8
1.4	Plan	9

This thesis aims at proposing a solution to automatically expand t-uples of a semantic class, the set, given a small subset of its members, the seeds, from large collections of semi-structured documents using the Web, which is a particular kind of a vital task of *Information Extraction* (IE). In this thesis, a semantic class is defined as a set of words or t-uples with similar meaning. It is a meaning or concept representation. It is challenging to develop an automatic, domain-independent and scalable solution with little linguistic knowledge requirement to extract t-uples or relations of different complexity (e.g., varied arity) from a huge corpus. Our solution is a minimally supervised approach, which only requires a small set of seeds of the target semantic class as input. The proposed solution is also integrated in a bootstrapping process to improve the performance.

1.1 Motivation

IE deserves great significance in the field of *Information Retrieval* (IR), which has been widely acknowledged because of the rapidly boom of information available.

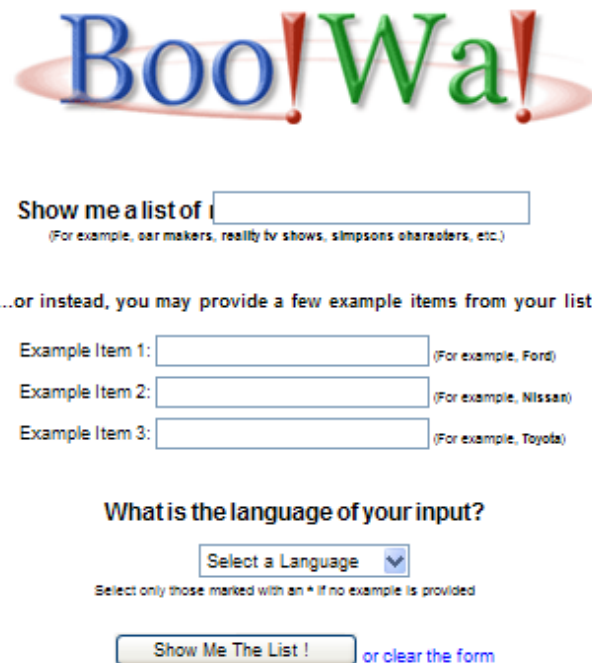
Its goal is to extract structured information of interest from unstructured and/or semi-structured documents.¹ As the goal hints, IE involves basically at least two categories according to the nature of data source, i.e. IE from unstructured data and IE from semi-structured data. In the first case, IE concerns mostly processing texts in human language, which requires techniques or tools of natural language processing (NLP). For the second case, in view of certain characteristics of semi-structured data, IE usually requires little linguistic knowledge. Instead certain structural information (e.g., tags) can be used to extract user-specified information. Among all the semi-structured data sources, the World Wide Web (WWW) is undoubtedly a best-known huge collection of semi-structured documents.

The World Wide Web is a vast repository of data on various aspects surrounding businesses, education, politics, sports, and so on. Our ability to browse and search through this vast amount of data to extract useful information has proved useful in many ways. Unfortunately, extracting meaningful information from the Web in an efficient way is a non-trivial problem. It is partly due to the fact that the data within the Web are largely unstructured and highly distributed. Nonetheless, because of its numerous applications to a wide variety of problems [Brin 1998, Badica 2005, Etzioni 2008, Kozareva 2008, Wang 2008], IE from the Web has received a considerable attention from the research community. The focus of this thesis is a particular technique for information extraction from the Web, which is commonly known as *Set Expansion* or *Relation Extraction*. Set expansion is important for many information retrieval and data mining tasks such as named entity recognition [Talukdar 2006], semantic lexicon induction [Igo 2009], open relation extraction [Etzioni 2008], hyponymy acquisition [Hearst 1992], and semantic class learning [Kozareva 2008], opinion mining [Zhang 2011].

¹In this thesis, we adopt a definition of IE, which only concerns extracting information from texts. Information extraction from multimedia is not in the scope of this thesis.

1.2 Set Expansion

The basic idea of set expansion is to extract elements of a particular semantic class from a given data source. More precisely, given a set of seeds (e.g., names) of a particular semantic class (e.g., ships or US presidents) and a collection of documents (e.g., HTML pages), the set expansion problem is to extract more elements of the particular semantic class from the collection of documents. Consider $\{Yuritamou, Salvor T, Towada\}$, and $\{George Washington, Ronald Reagan, Bill Clinton\}$ the names of cargo ships and US presidents, respectively, as sets of three seeds. The goal here is to extract the names of all the cargo ships and US presidents from the Web.



The image shows a web interface for Boo!Wa!. At the top, the logo "Boo!Wa!" is displayed in blue and green text with red exclamation marks. Below the logo is a main search form with the text "Show me a list of" followed by a text input field. A small note below the field says "(For example, car makers, reality tv shows, simpsons characters, etc.)". Below this is the text "...or instead, you may provide a few example items from your list." followed by three example input fields labeled "Example Item 1:", "Example Item 2:", and "Example Item 3:". Each field has a small note: "(For example, Ford)", "(For example, Nissan)", and "(For example, Toyota)" respectively. Below the example fields is the question "What is the language of your input?" followed by a dropdown menu labeled "Select a Language". A note below the dropdown says "Select only those marked with an * if no example is provided". At the bottom, there is a button labeled "Show Me The List !" and a link labeled "or clear the form".

Figure 1.1: Snapshot of Boo!Wa!

Boo!Wa!² is an existing set expansion system that works reasonably well in many cases. Figure 1.1 is a snapshot of Boo!Wa! website. As can be seen, there are three text fields which are used to accept atomic values (i.e., *seeds*) of a semantic

²<http://boowa.com/>

class as input. It is noted that it can only accept two or three atomic seeds. After clicking the button "Show Me The List !", it searches several Web pages that contain the given seeds on the Web, and analyze these pages to extract more candidates. Finally, through certain ranking mechanism, it will return a ranked list of candidates that tend to be of the same semantic class as that of the seeds. This site also offers two options to help the users to expand the set of seeds. One option is that users can specify the name of the semantic class in the text field after the label "Show me a list of" to filter potential *ambiguous* candidates. The other option is that users can specify of what language the seeds are. This option can be used to prune a huge collection of Web pages to be searched and analyzed on the Web, which are in different languages from that of the seeds. In this way, it improves the efficiency of the system.

...or instead, you may provide a few example items from your list.

Example Item 1: (For example, Ford)
 Example Item 2: (For example, Nissan)
 Example Item 3: (For example, Toyota)

What is the language of your input?

*English

Select only those marked with an * if no example is provided

Show Me The List ! or clear the form

Fetches and analyzed 3 webpages in 0.51 seconds.

#	Found Items
1	Golden Taka
2	Ocean Queen
3	Ebba Maersk
4	Black Watch
5	Gaz Horizon
6	Rosa Maersk
7	Kent Forest
8	Front Eagle
9	Ocean Alert
10	C S Victory
11	F W Horcham

Figure 1.2: Output of Boo!Wa!

To illustrate in a more detailed manner how Boo!Wa! works, let us consider



Figure 1.3: Snapshot of Google Sets.

the example of cargo ship mentioned before. . The input to the Boo!Wa! system is three cargo ship names (the seeds), i.e. $\{Yuritamou, Salvor T, Towada\}$. Using the seeds as keywords, it searches for the most relevant Web pages that contain the seeds. As highlighted in a round rectangular box in Figure 1.2, three Web pages that contain the given three cargo ships are fetched and analyzed to extract more candidate cargo ships. Through certain ranking mechanism (discussed in more detail in section 3.2.3), it returns a ranked list of candidate cargo ships, as illustrated in Figure 1.2. In this particular example, Boo!Wa! reported 3000 names (with many mentions that were not ships' names). In the US presidents case, Boo!Wa! reported most of the names.

Another well known system that does set expansion is Google Sets³. Figure 1.3 is a snapshot of Google Sets. As can be seen, there are five text fields which are used to accept atomic values (i.e., *seeds*) of a semantic class as input. Different from Boo!Wa!, Google Sets can accept one to five atomic values as seeds. When there is only one seed, the result sometimes can be a mixture or unpredictable if the seed

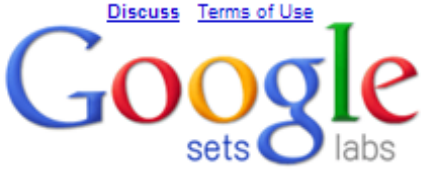
³<http://labs.google.com/sets>

is ambiguous (e.g., pear). Otherwise, it returns a list of atomic candidates of the same semantic class as that of the seeds. For the output, there are two choices of the size of the expanded set for the user, i.e. "Large Set" and "Small Set (15 items or fewer)". Even for "Large Set", Google Sets usually returns a set that is smaller than one hundred.

Since the technique used by Google Sets is proprietary, it is difficult to know how exactly it works. Thus, we can only examine its performance. Empirically, its performance may vary. In the case of cargo ships, it failed to report any results. Actually, using *Yuritamou* and/or *Salvor T* as seeds, it returns nothing. Using *Towada* as a seed, it returns a list of Japanese cities. This is because *Towada* is ambiguous and also refers to a city in Japan. Nonetheless, as expected Google Sets returned all the US presidents' names. Figure 1.4 shows part of the expanded set of US presidents.

In summary, existing set expansion systems work well for a given set of atomic seeds that unambiguously define a class. Generally, seeds can be represented by a set of *t-uples* or *relations* as in Codd's relational model. Like SEAL [Wang 2007] (which is actually the base of Boo!Wa!), some other proposals such as DIPRE [Brin 1998] mainly consider t-uples to be unary (i.e., sets of atomic values) or binary. A common framework adopted by many existing set expansion systems is based on a three-step method, as illustrated in Figure 1.5.

- Step One: Fetch relevant documents. Select a collection of documents containing the seeds, e.g. HTML pages collected from the Web using search engines, which may contain the keywords (seeds).
- Step Two: Construct patterns and extract candidates. Construct patterns (e.g., *wrappers* [Wang 2007]) from the seeds to extract candidate t-uples from the selected documents.
- Step Three: Rank candidates. Rank the candidate t-uples to find the most similar ones to the seeds, i.e. which are more likely to belong to the semantic



Discuss Terms of Use

Google
sets labs

Predicted Items
ronald reagan
george washington
bill clinton
george w bush
barack obama
richard nixon
jimmy carter
john f kennedy
abraham lincoln
george hw bush

Figure 1.4: Output of Google Sets.

class of the given seeds.

The main difference between various existing solutions lies in their different data source to expand given set of seeds, different strategies for constructing the patterns, and the ranking schemes. It is not in the scope of this thesis to discuss all the existing solutions. Rather we pay attention to the generalization of the problem, i.e. we depart from the expansion of the set of atomic values to the expansion of the set of t-uples for which the *arity* is greater than one.

The expansion of set of t-uples arises in many practical situations. Consider, e.g. the previous case of ships, now with the requirement of extracting not only the names but also the International Maritime Organization (IMO) numbers of the ships. That is, given the set $\{<Yuritamou, 9374076>, <Salvor T, 8618968>, <Towada, 9321213>\}$, expand it with more pairs of ships and their IMO numbers.

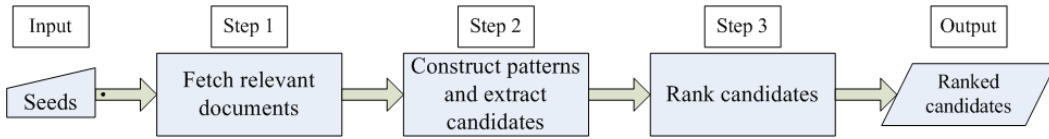


Figure 1.5: A three-step framework of set expansion systems.

Such expansions are needed for Schema Auto Completion (SAC) [Cafarella 2008, Elmeleegy 2009] in which IMO numbers may be needed (as primary keys to uniquely identify the ships) to perform certain operations. Intuitively, using a set of t-uples expansion scheme, the semi-structured data can be extracted from the Web to form lists, which can then be used (as input to a SAC solution such as the one proposed in [Elmeleegy 2009]) to populate relational tables.

1.3 Contributions

In this thesis, first, we argue that the set of t-uples expansion compels novel extensions to the existing solutions. While leveraging from the existing techniques we then propose an effective solution for set of t-uples expansion. To summarize, this thesis makes the following core contributions.

- We propose a regular expression based technique for making the wrappers more flexible that is more suitable for extracting candidates with higher arity, and hence more effective for the set of t-uples expansion (section 4.3.1).
- We propose a simple yet effective scheme for expanding the search to more pages, in particular to the collection of pages that belong to the same websites. This scheme allows discovering candidate t-uples not only from the pages that contain the seeds but also from their *sibling*⁴ pages that do not contain the seeds (section 4.3.2).
- We propose a new ranking scheme that takes into account the domains aim-

⁴By sibling Web pages we mean those Web pages that share a common domain or sub-domain.

ing at improving the ranking of the candidates (section 4.4). Our ranking scheme also facilitates the ranking of domains from which candidate t-uples are extracted. In other words we can check the *quality* of the domains that contributed in expanding the target set. To the best of our knowledge, none of the existing solutions provide this simple yet useful feature.

- We propose a bootstrapping process to improve the performance of our system (section 4.5).

A byproduct of our system is a ranked list of documents. It indicates the degree of relevance of a document to the given seeds and the target relation. We claim that such ranking makes much more sense than the ranking by frequency. Moreover, it has been verified in section 5.3. In the main body of this thesis, we present these contributions in detail.

1.4 Plan

This thesis is organized as follows. Chapter 2 summarizes some existing approaches that are related to our work to give a full picture of the research context of set expansion. In chapter 3, we provide the essential background of our work, i.e. DIPRE [Brin 1998] and SEAL [Wang 2007, Wang 2009], including architectures, algorithms and experimental results. In section 4.1, we first formulate the problem of set of t-uples expansion. Later in chapter 4 we present the details of our proposed set expansion system, especially the wrapper construction techniques and the ranking schema. We evaluate our proposals extensively while using several real datasets from the Web in chapter 5, and show the effectiveness of our proposed techniques. Finally, chapter 6 concludes the thesis and illustrates some directions on our future work.

Related Work

Contents

2.1 Taxonomy of Set Expansion Related Techniques	10
2.1.1 Taxonomy Based on Data Source	11
2.1.2 Taxonomy Based on Pattern Construction	12
2.1.3 Taxonomy Based on Arity of Seeds and Target Relations . . .	13
2.2 Representative Work	14
2.3 Comparison	16

In this chapter, we describe some research works that are related to the set expansion problem. We start by introduce a taxonomy of existing set expansion systems based on different metrics. For each category, we investigate its advantages and disadvantages. Thereafter, representative works of each category are summarized to offer more details. Finally, we conclude the differences between our work and the existing works. In this way, we aim to give the readers a full picture of the research context of the set expansion problem, and to explicitly locate the position of our work to make our contributions more clearly.

2.1 Taxonomy of Set Expansion Related Techniques

Set expansion problem has been studied under various names and forms [Talukdar 2006, Kozareva 2008, Wang 2008, Pantel 2009]. These proposals differ each other in the nature of data source (i.e., structured, semi-structured or unstruc-

tured; e.g., corpus or the Web), pattern constructions (e.g., distributional similarity, or wrapper induction), arity of seeds and target relations (i.e., unary, binary, or n-ary), and feature selections (i.e., semantic-level, syntactic-level, term-level or character-level). To make a systematic study of existing set expansion systems, we introduce a taxonomy based on abovementioned metrics. To start with, we describe the taxonomy based on the nature of data source.

2.1.1 Taxonomy Based on Data Source

From the point of view of data source, set expansion systems generally can be divided into two categories, i.e. corpus-based or Web-based. Typically, the former is designed to induce domain-specific semantic lexicons (e.g., proteins, genes) from a collection of domain-specific texts. Generally, it is easier to discover specialized terminology directly from a domain-specific corpus than from a broad-coverage corpus. Despite of that, accuracy may still be low because most corpuses are relatively small and adequate annotated or labeled data does not exist. However, as the word "Web" hints, the latter, typically, is designed to induce broad-coverage resources. It is challenging to find wanted specialized terminology because the Web is a vast and highly distributed repository of varied qualities and various granules.

Despite of different natures between corpus and the Web, researchers have proposed several set expansion systems based on the corpus and/or the Web. Firstly, the corpus-based set expansion systems usually require certain NLP techniques, such as parsing, Part-Of-Speech (POS) tagging, Named-Entity Recognition (NER), and etc.. Specifically, early corpus-based set expansion systems often use nouns co-occurrence statistics to extract lists of nouns with same properties, e.g. [Riloff 1997]. Later, some corpus-based set expansion systems start using syntactic relationships (e.g., *Subject-Verb* or *Verb-Object*) to extract sets of specific elements, e.g. [Widdows 2002]. There are also other well-known corpus-based systems which use lexicon-syntactic patterns (e.g., *such Noun as Noun list*) to find

user-specified relations, e.g. [Hearst 1992, Thelen 2002, Etzioni 2008]. Because of the requirement for parsing, POS tagging, or other linguistic knowledge, the above mentioned systems can only be evaluated on fixed corpus. Secondly, there also exist a couple of Web-based set expansion systems. Several Web-based systems are built on Hearst’s work [Hearst 1992], i.e. using hyponym patterns to extract candidate members of a semantic class, e.g. [Kozareva 2008]. Some Web-based systems discover candidate members of a semantic class using Web query logs (e.g., [Paşca 2007]). Many other systems many use the structural or URL information of Web pages to extract entities or relations of interest, e.g. [Brin 1998, Agichtein 2000, Crescenzi 2001, Badica 2004, Gilleron 2006, Wang 2007]. Moreover, there are also relation extraction systems that exploit the advantages of both corpus-based and Web-based techniques. For instance, Igo et al. in [Igo 2009] first expand a semantic lexicon from a domain-specific corpus, given a small set of its members. Then it computes the Pointwise Mutual Information (PMI) between the candidates and the seeds based on Web queries to filter the candidates.

2.1.2 Taxonomy Based on Pattern Construction

From the point of view of pattern constructions, set expansion systems generally can be divided into several categories, among which three most representative ones are Distributional Similarity (DS), Positive and Unlabeled examples Learning (PU Learning), and Wrapper Induction (WI). The DS approach is based on the distributional hypothesis that words of similar meanings tend to occur within similar context [Harris 1954]. Specifically, it first computes the surrounding word distribution of all the terms of interest including the given examples or *seeds*, usually through a context window and a feature vector. Thereafter, certain metric (e.g., TF-IDF, PMI) is adopted to compute a similarity score between vectors of the seeds and that of other terms to identify candidates. Moreover, this approach itself provides a ranking mechanism, which ranks the candidates according to this similarity

score, e.g. [Pantel 2009]. For the PU Learning, basically, it is a *binary-classification* problem. Specifically, given a set P of *positive* examples of a particular class and a set U of *unlabeled* examples, a classifier is trained using P and U for classifying the data in U or predicting the class of new arrival instances, e.g. [Li 2010]. Besides, the Bayesian Sets (e.g., [Ghahramani 2005, Zhang 2011]) can be considered as a special case of PU Learning. The minor difference lies in that PU Learning introduces an additional set—*Reliable Negative Set* to help train the classifier, except exploiting useful information in U . PU Learning is better than Distributional Similarity in that the former ranks the candidates not only through comparison with given seeds, but also using the information provided by other candidates. For the Wrapper Induction technique, it usually exploits character-level features and/or special structures (e.g., HTML tags) to identify candidates similar to the seeds, e.g. [Brin 1998, Crescenzi 2001, Badica 2005, Gilleron 2006, Wang 2008]. Generally, since it relies on certain structural information, it is not applicable to general free texts.

2.1.3 Taxonomy Based on Arity of Seeds and Target Relations

From the point of view of arity of seeds and target relations, many of existing systems have been developed for extracting atomic values (i.e., unary relation), e.g. [Thelen 2002, Widdows 2002, Paşca 2007, Wang 2008, Igo 2009, Pantel 2009]. Their tasks are either to build a semantic lexicon or to recognize certain named entities. There also exist several systems that aim to extract binary relations, e.g. [Brin 1998, Crescenzi 2001, Badica 2004, Mintz 2009, Wang 2009]. These systems use structural information or distant supervision to discover specific relations between pairs of entities. For the n -ary relation extraction, only a few solutions are proposed, e.g. [McDonald 2005, Gilleron 2006]. These systems are very complicated, and some even require interactions with users. In view of this, our goal of this thesis is to propose an automatic, effective solution to set of N -ary t -uples expansion.

2.2 Representative Work

To be more specific, several representative works that belong to the above set expansion taxonomy are summarized as follows. Talukdar et al. in [Talukdar 2006] induced a pattern automaton based on the term level feature to extract lists of named entities over a free text corpus. Mintz et al. [Mintz 2009] presented a *distant supervision* based solution for relation extraction. The basic idea underlying distant supervision is that any text fragment that contains a pair of entities comprising a binary relation in a well-known semantic corpus (e.g., Freebase) is likely to express that relation in a similar way. As can be seen, these two systems are corpus-based. Such systems works well for extracting low order relations, but not necessarily well for high order relations. McDonald et al. proposed a simple algorithm to extract high order relations in [McDonald 2005]. The main idea is to factor the high order relations into a set of binary relations and extract those binary relations to build an entity graph. High order relations are then constructed by finding maximal cliques in the entity graph.

For the Web-based systems, Kozareva et al. in [Kozareva 2008] used lexicon-syntactic patterns to extract hyponym lists from the Web. Etzioni et al. in [Etzioni 2004] developed a framework called KnowItAll which extracts entities or relations from the Web. The input to the framework is a small set of domain-independent, generic patterns and a set of names of semantic classes for the entities or relations to be extracted. The output is a list of entities or relations extracted from the Web. Etzioni et al. [Etzioni 2008] introduced an unsupervised extraction paradigm, Open Information Extraction, which extracts information without pre-defined relation-specific patterns via only a single pass over data. Based on this paradigm, they proposed TextRunner. It outputs a set of relations associated with a probability, which are indexed to support customized queries.

It is noted that these taxonomy criteria is not non-intersect. For instance, [Talukdar 2006] is a good example which adopts the DS approach as well.

Besides, Pantel et al. in [Pantel 2009] also proposed a distributional similarity based approach for automatic set expansion over Web-scale data. These approaches are language-dependent, since they construct patterns based on syntactic-level and/or term-level features, which requires NLP techniques such as parsing, POS tagging and etc..

In contrast to that Wang et al. proposed SEAL [Wang 2007], which is a language-independent system. The main idea of SEAL is to construct (character level) wrappers, which are used to extract suitable candidates from semi-structured data. Brin et al. proposed DIPRE [Brin 1998] for extracting a structured relation, e.g. $\langle author, book-title \rangle$ pairs from the Web. It exploits the redundancy within the contexts and duality between patterns and t-uples to extract the target relation. The main problem with DIPRE is that patterns are not flexible to extract candidates with high arity, and hence not very useful for the set of t-uples extraction. Agichtein et al. proposed Snowball in [Agichtein 2000], which tends to overcome the limitations of patterns in DIPRE. The key improvement of Snowball from the basic DIPRE is that the Snowball patterns introduce named-entity tags that are more effective for relation extraction.

Badica et al. in [Badica 2005] proposed an interesting approach L-wrappers that combines logic programming and information extraction. In their method inductive logic programming is used to extract binary relations from HTML documents. The main limitation of their method is that it does not work well for extracting high order relations. Crescenzi et al. [Crescenzi 2001] proposed a system called ROAD-RUNNER, which can automatically extract data from large websites given a set of sample HTML pages belonging to the same class. It is based on the theoretical background of *union-free regular expression*. Specifically, in order to induce a schema and extract data from the Web sites, it iteratively computes the least upper bounds on the RE lattice to generate a common wrapper of the input HTML pages. It is limited because it requires that all the HTML tags be known before hand, and that

the schema of the website be relatively simple. Besides, it is desired that the input Web pages be of the same class and of the same schema. It does not consider the cases where data records occur on a single page. As can be seen, the above systems, from SEAL to ROADRUNNER, are wrapper induction systems.

Schema Auto Completion (SAC) [Cafarella 2008, Elmeleegy 2009] and Word Sense Disambiguation (WSD) [Turdakov 2010] problems are basically different yet related to the set expansion problem. The main problem in SAC is to populate a relational table from a given list that is assumed to be extracted from the Web. Set expansion schemes could be important here to extract lists from the Web. The WSD problem is to find the *word-sense* (meaning within a context) of a given word by resolving the additional information provided with the particular word. Again, the resultant set of set expansion systems can be provided as a reference to help resolve the ambiguities in WSD problem.

2.3 Comparison

In this thesis, we aim to propose a minimally supervised set expansion system which constructs wrappers to extract a list of n-ary t-uples from the Web. Our work is different than the ones proposed in [Talukdar 2006, Kozareva 2008, Wang 2008, Pantel 2009], [Brin 1998, Agichtein 2000, Etzioni 2008, Mintz 2009] and [Cafarella 2008, Elmeleegy 2009] in many ways. In particular, all the approaches proposed in [Talukdar 2006, Wang 2007, Kozareva 2008, Pantel 2009] mainly deal with *atomic* set expansion or named-entity recognition. In contrast to that set of t-uples expansion is the main problem that we address in this thesis. [Agichtein 2000, Crescenzi 2001, Badica 2005, Gilleron 2006, Etzioni 2008, Mintz 2009] present solutions for *t-uple* or *relation* extraction. However, they either require certain linguistic knowledge or only work on documents with specific structures (or tags) or need to interact with the users. Besides, our approach for wrapper construction is different and flexible than the ones proposed in [Brin 1998, Wang 2009]. Moreover, our

system can automatically not only work on cases where multiple t-tuples occur on a single page, but also the cases where t-tuples appear on parallel Web pages (see section 4.3.2). We will explain these differences in detail in chapter 4.

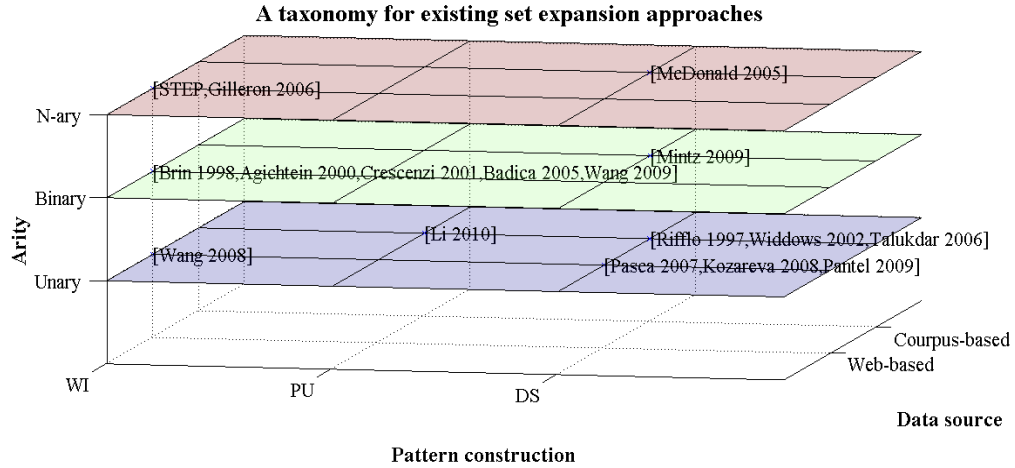


Figure 2.1: A taxonomy of set expansion related systems.

To obtain a full picture of the related literature, the above set expansion system taxonomy is visualized in Figure 2.1. This figure has three dimensions. Each corresponds to a metric for taxonomy. Specifically, the x-axis represents different ways of constructing patterns. There are three points along this axis, DS (Distributional Similarity), PU (Positive and Unlabeled examples Learning), and WI (Wrapper Induction). The y-axis represents for the nature of data source. Corpus-based and Web-based are two representative points along this axis. The z-axis describes the arity of seeds and target relation, along which there are three points, Unary, Binary and N-ary. We also draw three plates that correspond to three different arity of seeds and target relation. As can be seen from Figure 2.1, most of the existing systems extract unary or binary relations, which are under the plate $Arity = N - arity$. In this figure, one can easily locate the position of a set expansion or relation extraction system and then understand the research context of this topic. For instance, SEAL ([Wang 2007]) is a system which can induce wrappers based on a small set of examples of a semantic class to extract a list of atomic values of the same semantic

class from the Web. Hence, its coordinate in this figure is (*WI, Web-based, Unary*). Moreover, our proposed STEP is located at (*WI, Web-based, N-ary*).

SAC [Cafarella 2008, Elmeleegy 2009] is the problem of creating relational tables from the given lists. Our proposed techniques can be used as a pre-processing step for SAC. Besides, our work is also helpful for WSD. Specifically, the set of t-uples that we expand can also be used as a means of resolving ambiguity of certain t-uples caused by missing some attributes. As for the proposal in [McDonald 2005], we can use it to develop a set of t-uples expansion system over free text collections in the future.

Background

Contents

3.1	DIPRE	19
3.1.1	Step One: Fetch Relevant Documents	20
3.1.2	Step Two: Construct Patterns and Extract Candidates	21
3.1.3	Step Three: Rank Candidates	24
3.1.4	Performance Evaluation	24
3.2	SEAL	25
3.2.1	Step One: Fetch Relevant Documents	26
3.2.2	Step Two: Construct Patterns and Extract Candidates	27
3.2.3	Step Three: Rank Candidates	30
3.2.4	Performance Evaluation	31
3.2.5	Extend SEAL for Binary Relation Extraction	32

In this chapter, we review two set expansion systems that inspired our proposal, DIPRE ([Brin 1998]) and SEAL ([Wang 2007]). For each system, we first offer an overview of the system. Secondly, we will summarize the techniques they use step-by-step according to the three common steps illustrated in Figure 1.5. At the end, we will report some statistics of their performance.

3.1 DIPRE

Brin in [Brin 1998] addressed the problem of extraction relations from the World Wide Web. In the paper, he proposed a solution called Dual Iterative Pattern

Relation Expansion (DIPRE). The basic idea that underlies DIPRE is to exploit the duality between patterns and target relations.

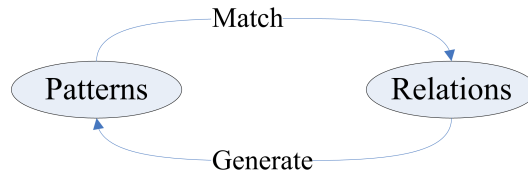


Figure 3.1: Duality between patterns and relations.

Specifically, as illustrated in Figure 3.1, given a set of good instances of target relations, a set of good patterns can be generated. Meanwhile, given a set of good patterns, the instances that match these patterns can be good candidates of target relations.

Author	Book-title
<i>Isaac Asimov</i>	<i>The Robots of Dawn</i>
<i>David Brin</i>	<i>Startide Rising</i>
<i>James Gleick</i>	<i>Chaos: Making a New Science</i>
<i>Charles Dickens</i>	<i>Great Expectations</i>
<i>William Shakespeare</i>	<i>The Comedy of Errors</i>

Table 3.1: Five seed books used in DIPRE [Brin 1998].

In this paper, the author considered a specific problem that extract more books from the Web given five $\langle author, book-title \rangle$ pairs as seeds, which is shown in Table 3.1 (from [Brin 1998]). Algorithm 1 (adapted from [Brin 1998]) illustrates how DIPRE works. Apparently, DIPRE pertains to the three-step framework in Figure 1.5. In the following, we will summarize the principles that DIPRE use in each step in turn.

3.1.1 Step One: Fetch Relevant Documents

This task is illustrated in line 3 in Algorithm 1. Firstly, DIPRE searches each Web page to find all the occurrences of all the seed pairs of author and book-title in text.

Algorithm 1: DIPRE's algorithm

```

Input:  $S, D$ ;
Output:  $R$ ;
1  $R = \emptyset$ ;
2  $R = R \cup S$ ;
   //Find occurrences of  $R$  in documents  $D$ 
3  $O = \text{FindOccurrences}(R, D)$ ;
   //Generate patterns  $P$  based on the occurrences of step3
4  $P = \text{GeneratePatterns}(O)$ ;
   //Apply the set of patterns  $P$  to extract a new set ( $R'$ ) of
   candidates of the target relation
5  $R' = \text{ExtractCandidates}(P, D)$ ;
6  $R = R \cup R'$ ;
7 if  $R$  is not large enough then
8   | Go to step 3;
9 return  $R$ ;

```

Specifically, it defines one occurrence of each seed pair as a 7-tuple, $\langle \text{author}, \text{book-title}, \text{order}, \text{url}, \text{prefix}, \text{middle}, \text{suffix} \rangle$. The *order* represents the order of the author and the book-title occurring on a Web page. For instance, let *order*=1 if the author appears before the book-title; otherwise *order*=0. The *url* is the Uniform Resource Locator (URL) of a Web page. The *prefix* is defined as the m characters preceding the author (or the book-title if the book-title is ahead of the author). Accordingly, the *suffix* consists of the m characters following the title (or the author). It is noted that m is a parameter that control the length of the left and right context of each occurrence. In the DIPRE paper, it is set to be 10. As for *middle*, it refers to the context between the author and the book-title. To be more specific, one example of an occurrence of the first seed book, i.e. $\langle \text{Isaac Asimov}, \text{The Robots of Dawn} \rangle$ is shown in Table 3.2.

3.1.2 Step Two: Construct Patterns and Extract Candidates

There are two subtasks in this step, i.e. pattern construction and candidate extraction. Pattern construction is the vital task in the entire information extraction process. This subtask corresponds to line 4 in Algorithm 1. In the paper [Brin 1998],

Attribute	Value
<i>author</i>	<i>Isaac Asimov</i>
<i>book-title</i>	<i>The Robots of Dawn</i>
<i>order</i>	1
<i>url</i>	http://www.ansible.co.uk/writing/shortrev.html
<i>prefix</i>	#asimov1">
<i>middle</i>	: <I>
<i>suffix</i>	</I></

Table 3.2: Example of an occurrence in DIPRE.

the author argued that since the Web is a broad-coverage repository, the patterns are sufficient if they have low false positive rate (i.e., patterns generating few incorrect pairs of author and book-title). Thus, patterns are constructed based on all the occurrences of the seed books. Specifically, DIPRE defines a 5-tuple pattern, $\langle order, urlprefix, prefix, middle, suffix \rangle$. Again, the *order* is a binary value to indicate the order of author and book-title.

Algorithm 2: GenerateOnePattern(O) (adapted from [Brin 1998]).

```

Input:  $O = \{o_1, o_2, \dots\}$ ;
Output:  $p = \langle order, urlprefix, prefix, middle, suffix \rangle$ ;
1 if ( $o_1.order = o_2.order = \dots$  &  $o_1.middle = o_2.middle = \dots$ ) is false then
2   return ;
3  $order = o_1.order$ ;
4  $middle = o_1.middle$ ;
   //Compute the longest common prefix of all the urls in  $O$ 
5  $urlprefix = LongCommonPrefix(\{o_1.url, o_2.url, \dots\})$ ;
   //Compute the longest common suffix of of all the prefixes in  $O$ 
6  $prefix = LongCommonSuffix(\{o_1.prefix, o_2.prefix, \dots\})$ ;
   //Compute the longest common prefix of all the suffixes in  $O$ 
7  $suffix = LongCommonPrefix(\{o_1.suffix, o_2.suffix, \dots\})$ ;
8 return  $p$ ;

```

Algorithm 2 (adapted from [Brin 1998]) illustrates how to generate one pattern based on all the occurrences of the seed pairs in DIPRE. The process of generating a pattern is as follows. First check whether the *order* and *middle* of all the occurrences are the same, respectively (line 1). If not, i.e. there does not exist a common *order* and/or a common *middle*, it is impossible to generate a pattern to match all the

seed books and the procedure returns none patterns. If so, there exists a potential pattern p . Set $p.order$ and $p.middle$ to the common $order$ and the common $middle$, respectively (line 3-4). It then computes the longest common prefix of all the urls of all the occurrences, and set $p.urlprefix$ to this common prefix (line 5). Similarly, find the longest common suffix of all the prefixes of all the occurrences, and the longest common prefix of all the suffixes of all the occurrences; and set them as $p.prefix$ and $p.suffix$, respectively (line 6-7). Overall, the 5-tuple $\langle order, urlprefix, prefix, middle, suffix \rangle$ is returned as a pattern.

It is noted that patterns generated by Procedure 2 can be too general, which extract a lot of non-books. To tackle this problem, DIPRE defines a metric called *specificity* to filter the patterns, which is given in Equation 3.1. Suppose p is a pattern, and $|s|$ is the length of s . Let n be the number of seed books whose occurrences are matched by the pattern p , and let t be a threshold. If and only if, one potential pattern p satisfies the Inequality 3.2, it can be considered as a pattern.

$$specificity(p) = |p.urlprefix| |p.prefix| |p.middle| |p.suffix| \quad (3.1)$$

$$n > 1 \ \& \ specificity(p) \times n > t \quad (3.2)$$

With Algorithm 2 as a subroutine and criteria *specificity* as a filter, it next proposes the Algorithm 3 (adapted from [Brin 1998]). Algorithm 3 first groups the occurrences by the *order* and *middle* (line 1). Then for each group, it calls Algorithm 2 to generate a pattern (line 3). If this potential pattern satisfies the specificity criteria in Eq. 3.2, it is considered as a real pattern (line 4-5). Otherwise, it separates the current group into subgroups according to the *url* attribute (line 7), and calls Algorithm 2 again to generate a pattern for each subgroup.

Once the patterns are generated, it comes to the next subtask, candidate extraction. For this subtask, it is relatively simple in DIPRE. For each pattern $\langle order,$

Algorithm 3: GeneratePatterns(O) (adapted from [Brin 1998]).

```

Input:  $O$ ;
Output:  $P$ ;
//Group all occurrences in  $O$  according to the order and the
middle.
1  $O' = \{O_1, O_2, \dots\} = \text{Group}(O)$ ;
2 foreach  $O_i \in O'$  do
3    $p = \text{GenerateOnePattern}(O_i)$ ;
4   if specificity( $p$ ) satisfies Eq. 3.2 then
5      $P = P \cup \{p\}$ ;
6   else
7     //Separate  $O_i$  into subgroups according to the url.
8      $O'_i = \{O_{i1}, O_{i2}, \dots\} = \text{SubGroup}(O_i)$ ;
9      $O' = O' \cup \{O'_i\}$ ;
10   $O' = O' \setminus \{O_i\}$ ;
10 return  $P$ ;

```

$\langle urlprefix, prefix, middle, suffix \rangle$, if the *order* is 1, and there is a document with a url matching the *urlprefix*, and a piece of text in this document matches the expression " $prefix[Author]middle[Book-title]suffix$ ", a candidate pair of $\langle author, book-title \rangle$ can be extracted.

3.1.3 Step Three: Rank Candidates

In DIPRE, the author does not propose any ranking approach. Thus, the final output is a set rather than a ranked list of pairs of author and book-title. Only generating patterns with very low false positive rate seems to be a compensation of the performance.

3.1.4 Performance Evaluation

In the experiment, DIPRE starts with the five books given in Table 3.1 over a part of the Stanford WebBase, which consists of 24 million Web pages amounting to 147 gigabytes. In the first iteration, only 199 occurrences of the five book pairs are discovered among the 24 million Web pages. Moreover, only three patterns

are generated based on the 199 occurrences. With the three patterns, it extracts 4,047 unique pairs of author and book-title. Using the 4,047 book pairs as seeds to run the second iteration, it collects 3,972 occurrences over about five million Web pages. As a result, 105 patterns, 24 of which have incomplete urls, are generated. In this iteration, 9,369 pairs of author and book-title are extracted over several million urls. Before starting the final iteration, 242 pairs of binary t-uples which have correct book-titles but with completely wrong authors are discarded manually. For the rest 9,127 books, it finds about 10,000 occurrences over roughly 156,000 Web pages. Consequently, these occurrences produce 346 patterns. A pass over the same repository generates 15,257 unique books. The number of seed books, number of documents searched from, number of occurrences and etc. in each iteration are summarized in Table 3.3.

Iteration	1	2	3
# seed books	5	4,047	9,127
# documents	24 million	5 million	156,000
# occurrences	199	3,972	9,938
# patterns	3	105	346
# resultant books	4,047	9,369	15,257

Table 3.3: Experimental statistics of DIPRE.

To evaluate, it randomly chooses twenty pairs of author and book-title from the 15,257 books. After manually checking the validation of the twenty books from the Web, nineteen out of them have correct book-titles.

3.2 SEAL

SEAL is proposed in [Wang 2007], short for "Set Expander for Any Language". As the name hints, it can expand sets of entities from a collection of semi-structured documents in *any* language. Similarly to DIPRE, SEAL constructs character-level wrappers as the maximally long common left and right context of give seeds, and then use such patterns to extract more candidates of the same semantic class as the

seeds. Actually, it is the way to construct character-level wrappers that contributes to its language-independence.

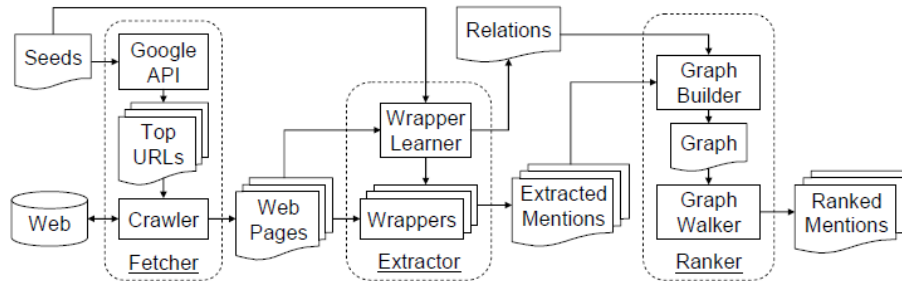


Figure 3.2: Flow chart of SEAL (from [Wang 2007]).

Similarly, in the following, we will give the details of SEAL according to the three-step framework in Figure 1.5. Moreover, it may be helpful to compare the flow chat of SEAL system in [Wang 2007], which is also given in Figure 3.2, with the three-step framework. As can be seen, there are three major components in SEAL system, i.e. Fetcher, Extractor and Ranker, which exactly correspond to the tasks of three steps in the framework 1.5. Firstly, let us consider the component Fetcher, also the first step.

3.2.1 Step One: Fetch Relevant Documents

As illustrated in Figure 3.2, it is the component Fetcher that accomplishes the task of fetching relevant documents. Specifically, the Fetcher uses the concatenation of all the seeds as keywords, and sends a query to Google search engine. A list of URLs of Web pages that contain the seeds will be returned. For example, given a set of cars as seeds, i.e. $\{Ford, Toyota, Nissan\}$, a snapshot of the top URLs returned by Google are shown in Figure 3.3. It is noted that all the top URLs contain all the seeds. It is more likely that there are other cars on these pages. For instance, another car named "Honda" appears on the top first Web page, which is highlighted in a rectangular box. Thus, the Web pages with the top URLs are downloaded to extract more candidates. A crawler is developed to download these Web pages.

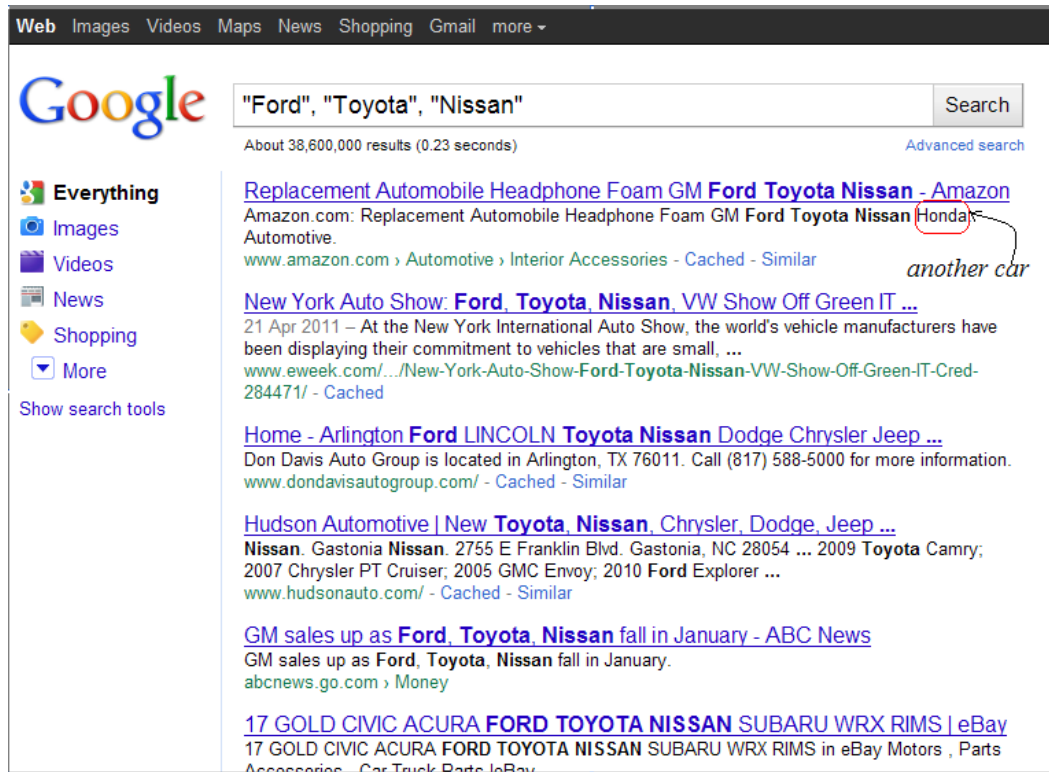


Figure 3.3: Top URLs containing "Ford", "Toyota" and "Nissan" returned by Google.

3.2.2 Step Two: Construct Patterns and Extract Candidates

For the second step, it is argued that the semi-structured Web pages have such characteristics that information within a same page is usually formatted consistently, but is quite different on different pages. Exploiting this characteristic of semi-structured pages, given a set of seeds, SEAL proposes a unsupervised approach to learn *wrappers* (i.e., page-specific extraction structures) for each page to extract candidates on the same page. In SEAL, the wrappers on a page is defined as the maximally long common left and right contexts surrounding the occurrences of seeds, at least one occurrence for each seed.

Given a set of seeds and a semi-structured page, the algorithm first locates all the occurrences of each seed on the page, and each occurrence is uniquely indexed with an *id*. For each occurrence of the seeds, its *left context* (i.e., all the characters

Wrappers MakeWrappers(Trie ℓ , Trie r)
 Return $\text{Wraps}(\ell, r) \cup \text{Wraps}(r, \ell)$

Wrappers Wraps(Trie t_1 , Trie t_2)
 For each $n_1 \in \text{TopNodes}(t_1, \ell)$
 For each $n_2 \in \text{BottomNodes}(t_2, n_1)$
 For each $n_1 \in \text{BottomNodes}(t_1, n_2)$
 Construct a new Wrapper($\text{Text}(n_1)$, $\text{Text}(n_2)$)
 Return a union of all wrappers constructed

Nodes BottomNodes(Trie t_1 , Node n')
 Find node $n \in t_1$ such that:
 (1) $\text{NumCommonSeeds}(n, n') == |\text{Seeds}|$, and
 (2) All children nodes of n (if exist) fail on (1)
 Return a union of all nodes found

Nodes TopNodes(Trie t , int ℓ)
 Find node $n \in t$ such that:
 (1) $\text{Text}(n).length \geq \ell$, and
 (2) Parent node of n (if exist) fails on (1)
 Return a union of all nodes found

String Text(Node n)
 Return the textual string represented by the path from root to n in the trie containing n

Integer NumCommonSeeds(Node n_1 , Node n_2)
 For each index $i \in \text{Intersect}(n_1, n_2)$:
 Find the seed at index i of seed instance list
 Return the size of the union of all seeds found

Integers Intersect(Node n_1 , Node n_2)
 Return $n_1.indexes \cap n_2.indexes$

Figure 3.4: Pseudo-code for wrapper construction of SEAL (from [Wang 2009]).

preceding this occurrence), and *right context* (i.e., all the characters following this occurrence) are inserted into a left context trie and a right context trie, respectively, where the left context is inserted in a reversed order. In the left context trie, each node maintains a list of *ids* which indicate the seed occurrences that follow the string associated with that node. Since the wrapper is defined as a pair of maximally long common left context and maximally long common right context that brackets at least one occurrence of each seed. Thus, the maximally long common left context is computed by a search over the left context trie for nodes that contain at least one *id* of each seed, and none of their children have this property. After that, for each of these longest strings, we find all the maximally long common right contexts in

the right context trie, and vice versa. Each pair of such maximally long common contexts is constructed as a wrapper. The pseudo-code for wrapper construction is illustrated in Figure 3.4 (from [Wang 2009]), where *Seeds* represents the set of input seeds and ℓ stands for the minimum length of the strings.

Once wrappers are constructed, they are used to match strings on the same page where the wrappers are constructed. Any strings bracketed by a wrapper are extracted as candidates or mentions (which is used in SEAL). From the way of wrapper construction, it verifies that SEAL is language-independent.

```

...<li class="first-of-type yuimenuitem">
<a class="yuimenuitemlabel" href="http://ford.dondavisautogroup.com/">
Ford LINCOLN</a></li><li class="yuimenuitem">
<a class="yuimenuitemlabel" href="http://nissan.dondavisautogroup.com/">
Nissan</a></li><li class="yuimenuitem">
<a class="yuimenuitemlabel" href="http://toyota.dondavisautogroup.com/">
Toyota</a></li></ul><ul class="hastitle CSSbyS CSRight">
<li class="yuimenuitem">
<a class="yuimenuitemlabel" href="http://dodge.dondavisautogroup.com/">
Dodge Chrysler Jeep Ram</a></li><li class="last-of-type yuimenuitem">
<a class="yuimenuitemlabel" href="http://scion.dondavisautogroup.com/">
Scion</a></li></ul></div>...

```

Table 3.4: HTML codes for a Web page.

Wrapper	
<i>Longest left context</i>	yuimenuitem"><a class="yuimenuitemlabel" href="http://
<i>Longest right context</i>	.dondavisautogroup.com/">
Candidates or mentions	
dodge, scion	

Table 3.5: One wrapper and two candidates on the Web page in Table 3.4.

Let us see an example. Again, we use the cars $\{Ford, Toyota, Nissan\}$ as seeds. Part of HTML codes for a Web page¹ returned by Google is given in Table 3.4, in which occurrences of seeds are marked in italic. According to the construction algorithm in Figure 3.4, one wrapper can be constructed and two candidates can be extracted using this wrapper on this page, which are summarized in Table 3.5.

¹<http://www.dondavisautogroup.com/>

3.2.3 Step Three: Rank Candidates

Another major contribution of SEAL is that it proposes a ranking mechanism using a graph model to rank extracted candidates. Generally, a graph is built to integrate all the entities and the relationships among them, for instance, seeds are used to find documents, wrappers can be derived from the documents, and mentions can be extracted by the wrappers. The nodes and relations between these nodes are summarized in Table 3.6 (from [Wang 2007]).

Source Node	Relation	Target Node
seeds	<i>find</i>	document
document	<i>derive</i> <i>find</i> ⁻¹	wrapper seeds
wrapper	<i>extract</i> <i>derive</i> ⁻¹	mention document
mention	<i>extract</i> ⁻¹	wrapper

Table 3.6: Nodes and relations in the graph in SEAL (from [Wang 2007]).

After the graph is built, it performs a lazy walk on this graph to measure the similarity between two nodes. Let x, y be nodes. If there is a binary relation r between x, y , it can be represented as $x \xrightarrow{r} y$. To walk away from a node x , it first uniformly picks a relation r , and then given r , uniformly picks a target node y . The two probabilities are given in the Equation 3.3 (from [Wang 2007]).

$$P(r | x) = \frac{1}{|r : \exists y x \xrightarrow{r} y|}; P(y | r, x) = \frac{1}{|y : x \xrightarrow{r} y|}; \quad (3.3)$$

In each lazy walk, it introduces a factor λ to indicate the probability of staying at x . Hence, the probability of walking away from x to z is recursively computed as follows (from [Wang 2007]).

$$P(z | x) = \lambda \cdot I(x = z) + (1 - \lambda) \sum_r [P(r | x) \sum_y P(y | r, x) P(z | y)]; \quad (3.4)$$

where $I(x = z)$ is a binary function, which returns 1 if node x and node z are a

same node, and returns 0 otherwise.

After enough iterations of lazy walk, each node will be assigned a weight, which stands for the probability of reaching this node in a random walk on this graph. And then it ranks all the nodes of the type "mention" by their weights.

3.2.4 Performance Evaluation

For the experiment, the authors collect 36 datasets in three languages, i.e. English, Chinese and Japanese, 12 datasets per language. The explanation of the 36 datasets is summarized in Table 3.7 (from [Wang 2007]).

	Dataset	Eng	Chi	Jap	Class Description
1	classic-disney	✓	✓	✓	Classic Disney movie names
2	constellations	✓	✓	✓	Constellation names
3	countries	✓	✓	✓	Country names
4	mlb-teams	✓	✓	✓	Major League Baseball team names
5	nba-teams	✓	✓	✓	National Basketball Association team names
6	nfl-teams	✓	✓	✓	National Football League team names
7	popular-car-makers	✓	✓	✓	*Popular car manufacturer names
8	us-presidents	✓	✓	✓	United States president names
9	us-states	✓	✓	✓	United States state names
10	cmu-buildings	✓			Carnegie Mellon's building names
11	common-diseases	✓			*Common disease names
12	periodic-comets	✓			Periodic comet names
13	china-dynasties		✓		Chinese dynasty names
14	china-provinces		✓		Chinese province names
15	taiwan-cities		✓		Taiwanese city names
16	japan-emperors			✓	Japanese emperor names
17	japan-priministers			✓	Japanese priminister names
18	japan-provinces			✓	Japanese province names

Table 3.7: Explanation for each dataset (* are incomplete sets) (from [Wang 2007]).

Moreover, it measures the performance by mean average precision (MAP), which is commonly used for evaluating ranked lists in IR. MAP combines both *recall* and *precision* aspects, and is simply the mean value of average precisions of multiple ranked lists. Suppose L is a ranked list, its average precision is defined as in Equation 3.5 (from [Wang 2007]).

$$AvgPrec(L) = \frac{\sum_{i=1}^{|L|} Prec(i) \cdot NewEntity(i)}{\# True Entities}; \quad (3.5)$$

where $Prec(i)$ is the precision at i . $NewEntity(i)$ is a binary function, which returns 1 if a) the extracted t-uple at i matches any true relation, and b) there exist no other extracted t-uples at rank less than i that is of the same relation as the one at i . It returns 0 otherwise.

In the experiments, for each dataset, the extraction in [Wang 2007] is an iterative process as follows.

- "1. Randomly select three true entities and use their first listed mentions as seeds.
2. Expand the three seeds obtained from step 1.
3. Repeat steps 1 and 2 five times.
4. Compute MAP for the five resulting ranked lists."

Besides, it collects the top 100, 200, 300 URLs returned by Google for each query. The MAP of the 36 datasets over the top 100, 200 and 300 URLs, achieves 93.13%, 94.03%, and 94.18%, respectively.

3.2.5 Extend SEAL for Binary Relation Extraction

Based on the basic SEAL, Wang et al. in [Wang 2009] extend it to extract binary relations. For the three components in SEAL, the extension from sets of atomic values expansion to set of binary relations expansion only arises problems in the second component. Thus, the vital task is to modify the wrapper construction algorithm given in Figure 3.4 to support binary relation extraction.

3.2.5.1 Construct Relational Wrappers

To make it work, it introduces another type of context, *middle* context, to describe the strings that occur between the two attributes of each binary t-uple. Specifically, given a set of seed pairs, the algorithm first locates their occurrences in the documents returned by Google. Thereafter, same as the original algorithm, the left context and right context are inserted into the left context trie and right context

trie. However, the middle context, together with a flag indicating whether the order of each occurrence is the same as the seed pair, is inserted into a list. An *id* maintained by a node indexes not only a seed occurrence but also a middle context.

In order to construct wrappers that bracket binary t-uples, the "Intersect" procedure in Algorithm 3.4 has to be rewritten as follows (from [Wang 2009]).

```
"Integers Intersect(Node  $n_1$ , Node  $n_2$ )
  Define  $S = n_1.indexes \cap n_2.indexes$ 
  Return the largest subset  $s$  of  $S$  such that:
    Every index  $\in s$  corresponds to the same middle context"
```

It returns all the seed pairs that are surrounded by the strings associated with two input nodes (i.e., n_1 , n_2) with the same middle context. Every *relational wrapper* consists of a pair of maximally long common left context and maximally long common right context, and an exactly matched middle context, which brackets at least one occurrence of each seed pair.

3.2.5.2 Performance Evaluation

Name	Attribute	Language	Size	Complete
US Governor	<US State, Governor>	<English, English>	56	Yes
Taiwan Mayor	<Taiwan City, Mayor>	<Chinese, Chinese>	26	Yes
NBA Team	<NBA Team, NBA Team>	<Chinese, English>	30	Yes
Federal Agency	<US Federal Agency Acronym, Full Name>	<English, English>	387	No
Car Maker	<Car Manufacturer, Headquarter>	<English, English>	122	No

Table 3.8: Five datasets for evaluating relational SEAL (adapted from [Wang 2009]).

In the experiment, five datasets of binary relations are manually collected, which are illustrated in Table 3.8 (adapted from [Wang 2009]).

For each dataset, it randomly chooses two seeds and bootstraps ten iterations. Again, it uses the MAP metric to evaluate the relational wrappers. The MAP of the five datasets achieves 89.2%.

STEP: Set of T-uples Expansion

Contents

4.1	Problem Formulation	35
4.2	Overview of STEP	36
4.2.1	Step One: Fetch Relevant Documents	37
4.2.2	Step Two: Construct Patterns and Extract Candidates	38
4.2.3	Step Three: Rank Candidates	39
4.3	Step Two: Construct Wrappers and Extract Candidates	40
4.3.1	Regular Expression Based Wrappers	40
4.3.2	Extracting T-uples from Sibling Pages	45
4.4	Step Three: Rank Candidates	51
4.5	Bootstrapping of STEP	55

In this chapter, we present our own approach, i.e. a minimally supervised framework for expanding a given set of t-uples, called STEP. Our STEP also pertains to the common three-step framework in Figure 1.5. Specifically, it starts with a small set of seed t-uples, which are then used to locate Web pages that contain the seeds on the Web. Next, regular expression based wrappers are constructed on the basis of the occurrences of seed t-uples on these pages. Consequently, all the suitable strings that match these wrappers are extracted as candidate t-uples. Finally, using certain ranking mechanism such as PageRank, all the candidate t-uples are ranked to produce a ranked list as the output. This chapter is organized as follows. We start with a formulation of the set of t-uples expansion problem and summarize

several potential challenges in section 4.1. Thereafter, an overview of our proposed system is illustrated in section 4.2. In the remaining sections, we give a detailed presentation of algorithms and techniques used in each component of STEP, which also corresponds to the common three steps in turn.

4.1 Problem Formulation

To be precise, we first formulate the set of t-uples expansion problem as follows.

Let \mathbf{D} be a collection of documents, \mathbf{S} be a semantic class, and $\mathbf{R} = \{r_1, r_2, \dots, r_{N_s}\}$ be a set of seed t-uples such that every seed t-uple of \mathbf{R} , r_i , belongs to the semantic class \mathbf{S} . The set expansion problem is to extract a target set, $\mathbf{R}' = \{r'_1, r'_2, \dots, r'_{N_c}\}$, from \mathbf{D} , such that every t-uple of \mathbf{R}' , r'_j , belongs to the same semantic class \mathbf{S} . (Note that we do not put restrictions on the size of the input and target sets, but usually $N_c \gg N_s$.)

As summarized in chapter 2, most of existing works focus on extracting atomic values or binary relations. The set expansion is relatively easy if the seeds and the target set consist of atomic values, i.e. when the arity of t-uples is 1. Despite of that, these systems, especially DIPRE and SEAL introduced in chapter 3, inspire us in some aspect, such as the character-level wrapper construction, entity graph modeling and etc.. On the basis of such background, we aim to extend the set of atomic values or binary relations expansion to the set of t-uples expansion. The generalization of the set expansion, however, raises new problems at every stage of the expansion process, mainly, location of the source documents, wrapper constructions for the extraction of the candidate t-uples, and the ranking of the candidate t-uples.

All these and other potential problems are primarily due to the fact that *parts* of a seed (recall that the seeds now have multiple attributes) may be located arbitrarily on a Web page, i.e. without exactly consistent structures such as tables between the values of multiple attributes. The situation becomes even worse when the arity

of seed t-uples increases. In a worse case¹ all the seeds may not be on one page, and rather on multiple sibling pages of a particular website. In this situation, there are two possible solutions that can be adopted: (1) Construct wrappers in such a way that they can extract t-uples (of multiple attributes) that are not necessarily in an exactly consistent form. (2) Locate the sibling pages of the pages that contain the seeds from a website whenever applicable. To fix these problems, we propose a system called Set of T-uples ExPansion (STEP). Before presenting these solutions, we first give an overview of our system.

4.2 Overview of STEP

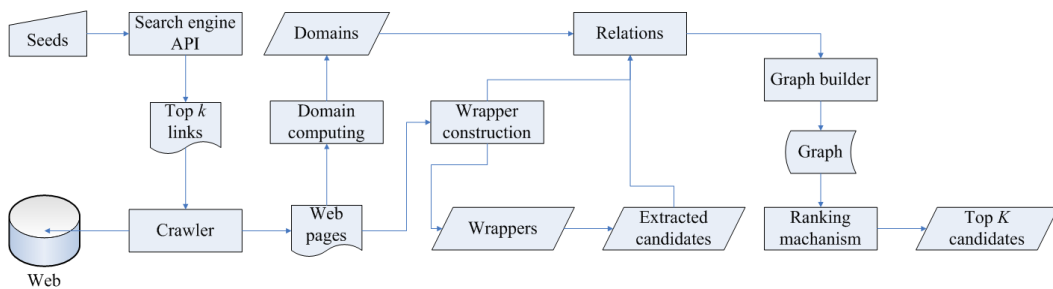


Figure 4.1: Architecture of STEP.

In this section, we present an overview of STEP, which is illustrated in Figure 4.1. It is very similar to that of SEAL in Figure 3.2. The difference lies in that we introduce a new node (*domain*) and set of new relations while building the graph. As a matter of fact, most set expansion systems have similar architectures, since they pertain to the common three-step framework in Figure 1.5. The major difference is in the way to develop a feasible approach to construct patterns, rank candidates and etc.. Again, we will describe STEP in three steps in the following.

¹In the worst case, even attributes of a single seed can be distributed over several Web pages. It is quite complicated and out of the scope of our current work. In the future, we will study further on this case.

4.2.1 Step One: Fetch Relevant Documents

Given a set of seed t-uples, STEP first forms a query, and submits it to search engines² to locate the Web pages that contain the seeds. STEP does not require any specific search engine. However, the *quality* of the Web pages returned by a specific engine will eventually affect the *quality* of the resultant list. Furthermore, a query to the search engines can be constructed in many ways, e.g. by grouping the corresponding attributes of the seed t-uples. Different ways to construct queries may result in different ranking of Web pages returned by a search engine. Hence, in turn it will impact the set of candidates to be extracted from these pages. Finally, it will affect the final ranking list. To be more clear, given a set of amateur radio magazines {<Amateur Radio, India>, <Funkamateur, Germany>} as the seeds, we make a query (i.e., query 1) which is of the same order of the seeds to Google, we collect the top five URLs in Table 4.1.

Top ID	Top URL
1	www.qrz.com/callsign/ik1pmr/
2	en.wikipedia.org/wiki/List_of_amateur_radio_magazines
3	www.ac6v.com/Magazine2.htm
4	www.enotes.com/topic/List_of_amateur_radio_magazines
5	www.rlx.lu/rl_ham_links.htm

Table 4.1: Top five URLs of query 1 returned by Google.

Top ID	Top URL
1	www.qrz.com/callsign/ik1pmr/
2	www.ac6v.com/Magazine2.htm
3	en.wikipedia.org/wiki/List_of_amateur_radio_magazines
4	www.rac.ca/ariss/arisstat.txt
5	cq-cq.eu/root.htm

Table 4.2: Top five URLs of query 2 returned by Google.

Besides, if we first group the seed t-uples by attributes, i.e. {{<Amateur Radio, Funkamateur>}, {<India, Germany>}} and then we make a query (i.e., query 2) to Google. The top five URLs returned by Google are summarized in Table 4.2. Comparing these two tables, the lists of top five URLs of different queries are different,

²We used popular Google and Yahoo! for this purpose.

for example, the top 2nd URL of query 1 becomes the top 3rd of query 2, and the top 5th URL of query 2 does not even exist in the top five URLs of query 1.

Given a set of seeds, how to make a query to return more relevant Web pages is another interesting problem. To simplify, we combined all the seed t-uples (without grouping their attributes) to form a query (i.e., the way same as query 1) in this thesis. In the future, we plan to study the impact of the order of attributes on the quality of results.

Moreover, except the order of attributes of the seed t-uples, the number of seeds, the arity of seeds and different choices of seeds will also have impact on the Web pages returned by search engines. Furthermore, the wrappers constructed on these pages and candidate t-uples extracted by these wrappers can be different. Consequently, the resultant ranking list will be different. These factors and their impact on the performance will be studied in section 5.3 in detail.

Intuitively, search engines can return a large number of pages for the queries submitted to them. Arguably, some of them may be irrelevant to the given queries. Moreover, search engines usually return pages that are already *ranked* according to the supplied query; therefore, it makes sense to use selective pages only. To that end, STEP uses the top N_p pages only from all the pages returned by the search engines. N_p is user-specified parameter, which controls the number of pages returned by a search engine. This parameter and its tuning will be studied in section 5.3 as well.

4.2.2 Step Two: Construct Patterns and Extract Candidates

Given the seeds and documents that contain the seeds, STEP first locates the occurrences of the seed on these documents. Based on these occurrences, it constructs wrappers. Then, these wrappers are used to extract candidate t-uples. For the wrapper construction, we find that the exactly matching mechanism used in DIPRE and SEAL are sometimes too restrictive, especially for n -ary t-uple extraction. Hence, we propose a *regular expression* based approach (section 4.3.1) to construct wrappers.

It is more flexible and suitable for high order relation extraction.

Besides, the wrapper construction of SEAL is based on the assumption that information within a same page is usually formatted consistently, but is quite differently formatted on different pages. Thus, it proposes page-specific wrappers. That is, the wrappers are used to extract candidates over the same pages where the wrappers were constructed. However, DIPRE seems to go into another extreme. It requires all the occurrences of the seeds over all different documents to appear in similar contexts to construct wrappers, despite that it introduces URLs to group Web pages to relax the constraint a little bit. In this thesis, our STEP is a compromise and combination of DIPRE and SEAL. That is, we do not only construct page-specific wrappers as SEAL to extract candidate t-uples from a single document, but also propose a way to extract candidate t-uples over sibling pages which is similar to DIPRE. The wrapper construction of STEP will be presented in detail in section 4.3.

4.2.3 Step Three: Rank Candidates

After obtaining the candidate t-uples, we consider rank them to distinguish the *good* candidates from the *spurious* ones. In this thesis, we use a *graph* model to rank the extracted candidate t-uples. Specifically, all the entities, such as seeds, Web pages, wrappers and etc., and the relationships between them are used to build an entity graph. Unlike SEAL, we introduce other entities, i.e. *domains*, as a new type of nodes in the entity graph. Apparently, a new set of relations or edges should be included to link this new type of nodes to other nodes in the graph. Based on this graph, we rank the candidates according to certain ranking mechanism (e.g., PageRank). Our ranking mechanism will be illustrated in section 4.4. Finally, the top N_c candidates are reported by STEP as output. N_c is also a user-specified parameter, which controls the number of top candidates returned by STEP. Next, we present the details of STEP while addressing these problems that arise due to

the generalization of the set expansion problem in step two and step three.

4.3 Step Two: Construct Wrappers and Extract Candidates

As discussed before, the way of wrapper construction in DIRPE and SEAL is limited for high order relation extraction. In this section, we propose a regular expression based way to construct wrappers which is more flexible and suitable for set of t-uples expansion. Besides, we observe that sometimes the given seeds are distributed on several pages from a same domain or sub-domain. Thus, we consider construct wrappers to extract t-uples over sibling pages. In the following, we will describe the two extensions in detail.

4.3.1 Regular Expression Based Wrappers

A wrapper generally consists of *contexts* surrounding the attributes of the given seeds and the candidate t-uples that are yet to be fetched. It implies that the wrapper becomes very complex when the arity of the t-uples increases. In DIPRE [Brin 1998], a wrapper can be generated only if it brackets all the occurrences of the seeds on the pages. It is a very strong constraint, which will decrease the recall dramatically. It has been proved by the fact that in the experiment of DIPRE, using five books as seeds, after a single pass over 24 million documents, only three patterns are generated. Hence, in SEAL [Wang 2007], the authors argue that it is more feasible to relax the constraints while constructing the wrappers. Specifically, a wrapper will be generated if it brackets at least one occurrence of each seed on a page. In this way, SEAL outperforms DIPRE, especially over the *recall* metric. However, it has other limitations. One major limitation in SEAL (also in DIPRE) is that candidate t-uples can only be extracted from the Web pages if a wrapper finds an exact match (EM) on the Web pages. This approach (i.e., EM) works well when the t-uples being extracted are atomic. However, when the arity of t-uples increases, the chance that

a wrapper finds an exact match on a given Web page decreases. Hence, SEAL fails to extract many t-uples that are potentially *good* candidates for the expansion of a given set. Shortly we will give an example to illustrate this case. Moreover, the experimental results in section 5.3 also support our claims.

To address this problem, we argue to construct wrappers based on regular expressions (RE). To be precise, given a set of seeds S and a document d that contains the seeds, first we locate the occurrences of the seeds. Each occurrence of a seed is a $(N+1)$ -tuple as follows. $\langle prefix, middle_1, middle_2, \dots, middle_{N-1}, suffix \rangle$; where the *prefix* represents all the characters preceding each occurrence, *suffix* represents all the characters following the occurrence, and $middle_i$ represents for the middle context between the i^{th} and the $(i+1)^{th}$ attributes of this occurrence. For each occurrence, we generate regular expressions for the potential digitals, white spaces and other regular symbols in each occurrence. This task is implemented in the Algorithm 4 (which is called later by the Algorithm 5).

Algorithm 4: FindOccurrenceOnOnePage(S, d).

Input: $S = \{s_1, s_2, \dots, s_{N_s}\}, d$;
Output: $O = \{O_1, O_2, \dots, O_{N_s}\}$;

- 1 $O = \emptyset$;
- 2 **foreach** $s_i \in S$ **do**
- 3 $O_i = FindOccurrence(s_i, d)$;
- 4 **if** $O_i = \emptyset$ **then**
- 5 **return** \emptyset ;
- 6 $O'_i = \emptyset$;
- 7 **foreach** $o_{ij} \in O_i$ **do**
- 8 $o'_{ij} = RegularExpression(o_{ij})$;
- 9 $O'_i = O'_i \cup \{o'_{ij}\}$;
- 10 $O = O \cup \{O'_i\}$;
- 11 **return** O ;

Afterwards, if there exist at least N_s occurrences in a document, one occurrence for each seed, such that

- 1) a nonempty longest common prefix *LCPrefix* can be computed for all their *prefix* entry,

2) a nonempty longest common suffix $LCSuffix$ can be computed for all their $suffix$ entry, and

3) a pair of longest common prefix $LCMiddlePrefix_i$ and longest common suffix $LCMiddleSuffix_i$ can be computed for all their $middle_i$ entry,

a (N+1)-tuple wrapper can be constructed as follows, $\langle LCPrefix, \langle LCMiddlePrefix_1, LCMiddleSuffix_1 \rangle, \dots, \langle LCMiddlePrefix_{N-1}, LCMiddleSuffix_{N-1} \rangle, LCSuffix \rangle$. The procedure for constructing regular expression based wrappers is illustrated in Algorithm 5.

Algorithm 5: GenerateWrappers(S, d).

Input: $S = \{s_1, s_2, \dots, s_{N_s}\}, d$;
Output: $W = \{\langle prefix, middle_1, middle_2, \dots, middle_{N-1}, suffix \rangle\}$;
1 $\{O_1, O_2, \dots, O_{N_s}\} = FindOccurrenceOnOnePage(S, d)$;
2 **foreach** $\langle o_1, o_2, \dots, o_{N_s} \rangle \in O_1 \times O_2 \times \dots \times O_{N_s}$ **do**
3 $LCPrefix =$
 $LongestCommonPrefix(\{o_1.prefix, o_2.prefix, \dots, o_{N_s}.prefix\})$;
4 **foreach** $i = 1; i < N; i ++$ **do**
5 $LCMiddlePrefix_i =$
 $LongestCommonPrefix(\{o_1.middle_i, o_2.middle_i, \dots, o_{N_s}.middle_i\})$;
6 $LCMiddleSuffix_i =$
 $LongestCommonSuffix(\{o_1.middle_i, o_2.middle_i, \dots, o_{N_s}.middle_i\})$;
7 $LCSuffix =$
 $LongestCommonSuffix(\{o_1.suffix, o_2.suffix, \dots, o_T.suffix\})$;
8 **if** $LCSuffix \neq empty$ & $LCPrefix \neq empty$ &
 $\forall LCMiddlePrefix_i, LCMiddleSuffix_i \neq empty$ **then**
9 $w = \langle LCPrefix, \langle LCMiddlePrefix_1, LCMiddleSuffix_1 \rangle, \dots, \langle$
 $LCMiddlePrefix_{N-1}, LCMiddleSuffix_{N-1} \rangle, LCSuffix \rangle$;
10 $W = W \cup \{w\}$;
11 **return** W ;

To better understand this wrapper construction technique, consider a set consisting of two pairs of amateur radio magazines and their countries of origin as the seeds: $\{\langle Amateur\ Radio, India \rangle, \langle Funkamateurl, Germany \rangle\}$. Figure 4.2 shows a snapshot of one specific Web page³ returned by a search engine, which contains a list of amateur radio magazines. Table 4.3 illustrates part of the HTML source

³http://en.wikipedia.org/wiki/List_of_amateur_radio_magazines

```

—
<td>1932-present</td>
</tr>
<tr>
<td><i>Amateur Radio</i></td>
<td><span class="flagicon">&#160;</span><a
href="/wiki/India">India</a></td>
<td>English</td>
<td>Quarterly</td>
<td></td>
</tr>
<tr>
<td><i>Break In</i></td>
<td><span class="flagicon">&#160;</span><a href="/wiki/New_Zealand">New Zealand</a></td>
<td>English</td>
<td>Bimonthly</td>
<td>1927-present</td>
—
<td>Monthly</td>
<td></td>
</tr>
<tr>
<td><i>Funkamateur</i></td>
<td><span class="flagicon">&#160;</span><a
href="/wiki/Germany">Germany</a></td>
<td>German</td>
<td>Monthly</td>
<td></td>
</tr>
<tr>
<td><i>Hagal</i></td>
<td><span class="flagicon">&#160;</span><a
href="/wiki/Israel">Israel</a></td>
<td>Hebrew</td>
<td>5-6x per year</td>
<td></td>
</tr>
<tr>
—

```

Table 4.3: Demonstration of wrapper construction on a Web page.

Magazine	Country	Language	Frequency	Published
<i>73</i>	 United States	English	Monthly	1960-2003
<i>Amateur Radio</i>	 Australia	English	Monthly	1932-present
<i>Amateur Radio</i>	 India	English	Quarterly	
<i>Break In</i>	 New Zealand	English	Bimonthly	1927-present
<i>CQ Amateur Radio</i>	 United States	English	Monthly	1945-present
<i>CQ DL</i>	 Germany	German	Monthly	
<i>CQ ham radio</i>	 Japan	Japanese	Monthly	1946-present
<i>CQ-PA</i>	 Netherlands	Dutch	Monthly	1951-present
<i>CQ-QSO</i>	 Belgium	Dutch, French	Bimonthly	
<i>CQ VHF Magazine</i>	 United States	English	Quarterly	
<i>Electric Radio</i>	 United States	English	Monthly	1989-present
<i>Electron</i>	 Netherlands	Dutch	Monthly	1945-present
<i>Electronics Illustrated</i>	 United States	English	Monthly	1959-1961
<i>Five Nine</i>	 Japan	Japanese	Monthly	
<i>Funkamateur</i>	 Germany	German	Monthly	
<i>Hagal</i>	 Israel	Hebrew	5-6x per year	
<i>HamFormosa</i>	 Taiwan	Chinese	Monthly	

Figure 4.2: Snapshot of a Web page containing amateur radio magazines.

code for this page, in which one occurrence of the seed t-uples is written in italic type. Apparently, if we use exact match (EM) as performed by SEAL and DIPRE, no wrapper can be constructed from this specific Web page. As a consequence, no candidate t-uples can be extracted from this Web page either. However, if we define the middle part of a wrapper as of a pair of regular expressions of the maximally long common prefix and suffix, we can construct a wrapper, which is flexible and potentially more suitable for extracting candidate t-uples that otherwise cannot be extracted. Indeed that is the case in this particular example. A (2+1) t-uple wrapper, i.e. $\langle \textit{suffix}, \textit{middle}_1, \textit{prefix} \rangle$, is shown in Table 4.4. Once a wrapper is obtained, it is applied to the same Web page (from which the wrapper was constructed) to extract candidate t-uples. In this example this wrapper in Table 4.4 produces two other magazine pairs, i.e. $\langle \textit{Break In}, \textit{New Zealand} \rangle$ and $\langle \textit{Hagal}, \textit{Israel} \rangle$ (shown in bold in Table 4.3).

As can be seen, the way we construct wrappers does not require any *a priori*

<i>prefix</i>	</td> </tr> <tr> <td><i>
<i>middle</i> ₁	(</i></td> <td>{0,200}>(>)
<i>suffix</i>	</td> <td>

Table 4.4: An example of wrapper. Candidate t-uples occurring in the form of "*prefix*[Magazine Name]*middle*₁[Country]*suffix*" are extracted by this wrapper from the page shown in Table 4.3.

knowledge about the HTML tags or specific structures, which is different from the way of wrapper construction in [Crescenzi 2001, Badica 2005]. Besides, we require none interaction with the users such as in [Gilleron 2006] either. Actually, given a set of seeds, our proposal can automatically work on arbitrary semi-structured documents such as XML, XHTML and etc..

4.3.2 Extracting T-uples from Sibling Pages

It is worth recalling that the wrapper construction in SEAL is based on the hypothesis that the format of information is consistent within a single page, but quite different on different pages. Thus, it uses all the seed t-uples to construct page-specific wrappers for every selected Web page that is reported by the search engine. This technique works well as long as the search engine reports enough number of pages, each of which contains all the seed t-uples. However that may not necessarily be the case always, especially if no single Web page contains all the seed t-uples. Nonetheless, candidate t-uples may still exist, not on a single Web page, rather on sibling Web pages. So the problem here is to find those sibling Web pages that possibly contain candidate t-uples. Consider, e.g. $\{ \langle \textit{Aeolian Vision}, 9483554 \rangle, \langle \textit{Aegean Star}, 7502942 \rangle \}$, a set of instances of binary relation $\langle \textit{Ship Name}, \textit{IMO} \rangle$. A quick check with these seeds reveals that the information about various ships indeed exists on sibling Web pages of a domain, e.g. marinetraffic, shipspotting, and vesseltracker.

Seed	URL
<Aeolian Vision, 9483554>	www.marinetraffic.com/ais/shipdetails.aspx?mmsi=241085000
<Aegean Star, 7502942>	www.marinetraffic.com/ais/shipdetails.aspx?mmsi=239868000

Table 4.5: Two sibling pages from "marinetraffic.com".

Table 4.5 illustrates two sibling pages for the two given seed t-uples, respectively. The two pages are both from the same subdomain, i.e. "www.marinetraffic.com/ais/", with the common prefix of URL "www.marinetraffic.com/ais/shipdetails.aspx?mmsi=", and the format of the two pages are similar to each other. There exist a large set of Web pages from the same subdomain with the same common prefix of URL, and each of these pages contains a ship like the given seeds. Thus, we suspect that the following procedure is feasible for extracting t-uples over sibling pages.

- 1) Given a set of seeds, find a set of Web pages (called seed pages) such that each seed occurs on a Web page, and all these pages are from the same websites and with a common prefix of URLs.
- 2) Induce a regular wrapper based on the set of seeds and the set of seed pages,
- 3) Collect all the other sibling pages from the same domain and with the same common prefix of URLs,
- 4) Apply the regular wrappers obtained in step 2 to all these sibling pages collected in step 3 to extract candidate t-uples,
- 5) Repeat the extraction process through step 1 to step 4 for each domain or website where a set of seed pages can be discovered.

It is observed that there are many such cases as the ship example. It is essential to enable our STEP to extract t-uples from sibling pages. Therefore, we devise a scheme for such situation. Figure 4.3 illustrates the flow chat of this schema.

Specifically, we use a seed t-uple from the given seeds as a query to a search engine to collect the top N_p Web pages. We repeat this process for all the seeds. Afterwards, we group Web pages (i.e., seed pages) together by checking the domain

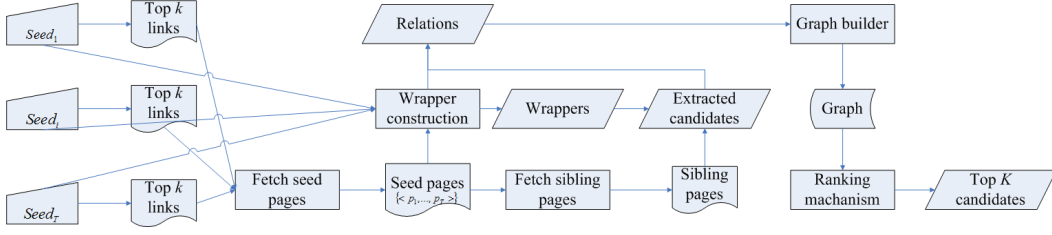


Figure 4.3: Schema for extracting t-uples from sibling pages.

of their URLs. For each URL group, we calculate their maximally long common prefix (of URLs), and use this prefix as a query to search engines. We again collect top N_p Web pages, which are intuitively sibling Web pages. Now by using the wrappers constructed based on the seed t-uples and the corresponding seed pages, those sibling Web pages are scanned for candidate t-uples. This process is repeated for every URL group. It is trivial to see that due to the sibling pages based scheme, STEP can discover pages that do not necessarily contain the seeds, but still may contain t-uples that are good candidates for the target relation. In the following, we will explain several important procedures in the schema 4.3 in detail.

Procedure FetchSeedPages($N_p, Seeds$)

Input: $N_p, Seeds = \{s_1, s_2, \dots, s_{N_s}\};$
Output: $SeedPages = \{ \langle p_1, p_2, \dots, p_{N_s} \rangle \};$

- 1 $SeedPages = \emptyset;$
- 2 **foreach** $s_i \in Seeds$ **do**
- 3 $L_i = \{l_{i1}, l_{i2}, \dots, l_{iN_p}\} = Find(s_i, N_p);$
- 4 **foreach** $\langle p_1, p_2, \dots, p_{N_s} \rangle \in L_1 \times L_2 \times \dots \times L_{N_s}$ **do**
- 5 **if** $Domain(p_1) = Domain(p_2) = \dots = Domain(p_{N_s})$ **then**
- 6 $SeedPages = SeedPages \cup \{ \langle p_1, p_2, \dots, p_{N_s} \rangle \};$
- 7 download all pages included in $SeedPages;$
- 8 **return** $SeedPages;$

Firstly, we will describe Procedure [FetchSeedPages](#). It illustrates the process for fetching seed pages in schema 4.3. Each input seed is used as a query to collect the top N_p pages from the search engines (lines 2 to 3). Thereafter, pages for different seeds that are from the same domain, one page for one seed, are grouped as seed

pages (lines 4-6). Finally, all the groups of seed pages are downloaded from the Web.

Algorithm 6: FindOccurrenceOnSiblingPages(S, D).

Input: $S = \{s_1, s_2, \dots, s_{N_s}\}, D = \{d_1, d_2, \dots, d_{N_s}\};$
Output: $O = \{O_1, O_2, \dots, O_{N_s}\};$

```

1  $O = \emptyset;$ 
2 foreach  $s_i \in S$  do
    //Locate occurrences of a seed on its corresponding seed page.
3    $O_i = FindOccurrence(s_i, d_i);$ 
4   if  $O_i = \emptyset$  then
5     return  $\emptyset;$ 
6    $O'_i = \emptyset;$ 
7   foreach  $o_{ij} \in O_i$  do
8      $o'_{ij} = RegularExpression(o_{ij});$ 
9      $O'_i = O'_i \cup \{o'_{ij}\};$ 
10   $O = O \cup \{O'_i\};$ 
11 return  $O;$ 

```

After obtaining the seed pages for each seed, the next procedure, which is a vital process of set expansion, is to construct wrappers. Generally, wrapper construction over a set of seed pages is similar to that over a single page. The minor difference lies in the way to locate occurrences of the seed. That is, instead of discovering occurrences of all the seeds on a single page, occurrences of a seed are located on its corresponding seed page. Hence, the input to the procedure to collect occurrences is a set of seeds and a set of corresponding seed pages. Thereafter, the Algorithm 4 can be easily revised as follows (see Algorithm 6).

Consequently, to construct wrappers over sibling pages, the only change of Algorithm 5 is to replace the first line with a call of Algorithm 6. The pseudo code of wrapper construction over sibling pages is shown in Algorithm 7.

Once the wrappers are constructed, the following step is to extract candidate t-uples from the sibling pages. Thus the whole process of extracting candidate t-uples from the sibling pages is illustrated in Procedure [ExtractOverSiblingPages](#). Specifically, it invokes the procedure described in Procedure [FetchSeedPages](#) to obtain the

Algorithm 7: GenerateWrappersOverSiblingPages(S, D).

Input: $S = \{s_1, s_2, \dots, s_{N_s}\}$, $D = \{d_1, d_2, \dots, d_{N_s}\}$;
Output: W ;

- 1 $\{O_1, O_2, \dots, O_{N_s}\} = \text{FindOccurrenceOnSiblingPages}(S, D)$;
- 2 **foreach** $\langle o_1, o_2, \dots, o_{N_s} \rangle \in O_1 \times O_2 \times \dots \times O_{N_s}$ **do**
- 3 $LCPrefix =$
 $\text{LongestCommonPrefix}(\{o_1.prefix, o_2.prefix, \dots, o_{N_s}.prefix\})$;
- 4 **foreach** $i = 1; i < N; i++$ **do**
- 5 $LCMiddlePrefix_i =$
 $\text{LongestCommonPrefix}(\{o_1.middle_i, o_2.middle_i, \dots, o_{N_s}.middle_i\})$;
- 6 $LCMiddleSuffix_i =$
 $\text{LongestCommonSuffix}(\{o_1.middle_i, o_2.middle_i, \dots, o_{N_s}.middle_i\})$;
- 7 $LCSuffix =$
 $\text{LongestCommonSuffix}(\{o_1.suffix, o_2.suffix, \dots, o_{N_s}.suffix\})$;
- 8 **if** $LCSuffix \neq \text{empty} \ \& \ LCPrefix \neq \text{empty} \ \&$
 $\forall LCMiddlePrefix_i, LCMiddleSuffix_i \neq \text{empty}$ **then**
- 9 $w = \langle LCPrefix, \langle LCMiddlePrefix_1, LCMiddleSuffix_1 \rangle, \dots, \langle$
 $LCMiddlePrefix_{N-1}, LCMiddleSuffix_{N-1} \rangle, LCSuffix \rangle$;
- 10 $W = W \cup \{w\}$;
- 11 **return** W ;

Procedure ExtractOverSiblingPages($N_p, N, Seeds$)

Input: $N_p, N, Seeds = \{s_1, s_2, \dots, s_{N_s}\}$;
Output: $Candidates = \langle attr_1, attr_2, \dots, attr_N \rangle$;

- 1 $Candidates = \emptyset$;
- 2 $SeedPages = \text{FetchSeedPages}(N_p, Seeds)$;
- 3 **foreach** $\langle p_1, p_2, \dots, p_{N_s} \rangle \in SeedPages$ **do**
- 4 $SiblingPages = \emptyset$;
- 5 $\{wrappers\} = \text{GenerateWrappers}(Seeds, \{p_1, p_2, \dots, p_{N_s}\})$;
- 6 **if** $\{wrappers\} \neq \emptyset$ **then**
- 7 $urlpre = \text{CommonPrefix}(p_1, p_2, \dots, p_{N_s})$;
- 8 $domain = \text{Domain}(p_1)$;
- 9 $SiblingPages = \{p_1, p_2, \dots, p_{N_p}\} = \text{Find}(urlpre \text{ site:domain}, N_p)$;
- 10 **foreach** $w_i \in \{wrappers\}$ **do**
- 11 **foreach** $p_j \in SiblingPages$ **do**
- 12 $\{candidates\} = \text{Extract}(w_i, p_j)$, where $\forall c \in \{candidates\}$,
- 13 $c = \langle attr_1, attr_2, \dots, attr_N \rangle$;
- $Candidates = Candidates \cup \{candidates\}$;
- 14 **return** $Candidates$;

seed pages (line 2). For each group of the seed pages, it constructs *wrappers* based on the occurrences of the seeds on the corresponding seed pages (line 5). If such *wrappers* exist, it fetches more sibling pages based on the maximally long common prefix of urls of this seed page group from the same domain (lines 7-9). Then it extracts candidate t-uples from these sibling pages based on the constructed *wrappers* (lines 10-13).

Parameters and procedures used in the Procedure `FetchSeedPages`, `ExtractOverSiblingPages` (and Procedure `BuildGraph` presented later) are summarized in Tables 4.6 and 4.7, respectively.

Parameter	Description
N	arity of seeds and candidate t-uples.
N_c	number of top candidate t-uples.
N_p	number of top pages returned by a search engine.
N_s	number of seed t-uples.
<i>siblingFlag</i>	a boolean flag indicating whether to extract candidates from sibling pages or not.

Table 4.6: Parameters description.

Procedure	Description
<i>Domain(link)</i>	a procedure that returns the domain of a given <i>link</i> .
<i>Extract(wrapper, page)</i>	a procedure that extracts and returns <i>candidates</i> bracketed by <i>wrapper</i> on <i>page</i> , i.e. $\{candidates\} = Extract(wrapper, page)$.
<i>Find(keywords, N_p)</i>	a procedure which sends a query (i.e., <i>keywords</i>) to a search engine and returns the top N_p pages that contain the <i>keywords</i> , i.e. $\{pages\} = Find(keywords, N_p)$, and $ \{pages\} = N_p$.
<i>CommonPrefix({link_i})</i>	a procedure that computes the maximally long common prefix <i>urlpre</i> of a set of links, i.e. $urlpre = CommonPrefix(\{link_i\})$.

Table 4.7: Procedures used in the Procedure `FetchSeedPages`, `ExtractOverSiblingPages`, and `BuildGraph`.

4.4 Step Three: Rank Candidates

Once the candidate t-uples are retrieved, the next task is to rank them in order to choose the most suitable t-uples that are semantically *close* to the seed t-uples. In their proposal Wang et al. in [Wang 2008] experimented with four ranking methods: (1) Random Walk with Restart, (2) PageRank, (3) Bayesian Sets, and (4) Wrapper Length. In our study, we implemented these algorithms as well. Experimental results from [Wang 2008] reveal that the PageRank method performed quite competitively. Therefore, in our evaluation chapter 5, we will only show the results of the PageRank method.

PageRank method works by constructing a graph in which all the entities, i.e. candidate t-uples, wrappers and the corresponding pages, are represented as nodes. An edge exists between two nodes if they have a *relation* between them, e.g. if a candidate t-uple c is extracted using a particular wrapper w , then there is an edge between c and w . Note that the domains (to which the Web pages belong) are not necessarily included in the graphs used in [Wang 2008]. We argue to include the domains in the graphs because the *relations* that are exploited to rank the candidate t-uples are less meaningful without representing the domains in the graphs. This change warrants an additional set of edges in the graph to establish relations between *domain* nodes and existing nodes. Overall, there are five different types of nodes and eight different kinds of relations between the nodes, which are summarized in Table 4.8.

Procedure `BuildGraph` summarizes the procedure that generates a graph while extracting t-uples. Furthermore, the candidate t-uples are ranked using the generated graph. A graph is initialized with the seed nodes (line 2). Pages are discovered on the Web based on the seeds (line 3). For every page found, the corresponding domain is fetched, and then inserted into the graph as nodes (lines 6-7). Edges between the current page and its domain are added (line 8). Edges are added between the page and each seed at lines 9-10. The regular expression based wrappers are

Source Node	Relation	Target Node
<i>Seeds</i>	<i>FIND</i>	<i>Pages</i>
<i>Pages</i>	<i>DERIVE</i>	<i>Wrappers</i>
<i>Pages</i>	<i>BELONGTO</i>	<i>Domains</i>
<i>Wrappers</i>	<i>EXTRACT</i>	<i>Candidates</i>
<i>Pages</i>	<i>INV_FIND</i>	<i>Seeds</i>
<i>Wrappers</i>	<i>INV_DERIVE</i>	<i>Pages</i>
<i>Domains</i>	<i>INV_BELONGTO</i>	<i>Pages</i>
<i>Candidates</i>	<i>INV_EXTRACT</i>	<i>Wrappers</i>

Table 4.8: The nodes and their relations in the graph.

constructed on the current page as shown in line 11. If wrappers exist on the current page (line 12), wrappers and edges between the current page and the wrappers are added (lines 14-15). If candidate t-uples are found (lines 16-17), they are added into the graph along with the edges between the wrappers and the candidate t-uples (lines 18-20).

Besides, Algorithm [ExtractOverSiblingPages](#) can be easily rewritten into a procedure (i.e., Algorithm 8) returning a graph, which contains all the entities and relationships between these entities while extracting candidate t-uples over sibling pages. Thus, we can use a boolean flag (e.g., *siblingFlag*) to determine whether to extract t-uples over sibling pages or not (lines 21-23).

Using the same seeds as in section 4.3.1, i.e., $\{\langle \textit{Amateur Radio}, \textit{India} \rangle, \langle \textit{Funkamateur}, \textit{Germany} \rangle\}$, part of the entity graph generated after running STEP is illustrated in Figure 4.4. To simplify, we only include one directional edge between each pair of nodes. As a matter of fact, each directional edge in the graph also has an inverse directional edge, as summarized in Table 4.8. Specially, we use the seed $\langle \textit{Amateur Radio}, \textit{India} \rangle$ as a query to a search engine, and find a document *Doc#1*. Two edges between the seeds and the document should be inserted into the entity graph, i.e. $\{\langle \langle \textit{Amateur Radio}, \textit{India} \rangle, \textit{FIND}, \textit{Doc\#1} \rangle, \langle \textit{Doc\#1}, \textit{INV_FIND}, \langle \textit{Amateur Radio}, \textit{India} \rangle \rangle\}$.

After the graph is generated, an iterative process is run to compute a PageRank

```

Procedure BuildGraph( $N_p, N, Seeds$ )


---


  Input:  $N_p, N, Seeds = \{s_1, s_2, \dots, s_{N_s}\}$ ;
  Output:  $G = \{V, E\}$ ;
  1  $V = \emptyset, E = \emptyset$ ;
  2  $V = V \cup Seeds$ ;
  3  $Pages = Find(Seeds, N_p)$ , where  $Pages = \{p_1, p_2, \dots, p_{N_p}\}$ ;
  4 foreach  $p_i \in Pages$  do
  5    $V = V \cup \{p_i\}$ ;
  6    $d = Domain(p_i)$ ;
  7    $V = V \cup \{d\}$ ;
  8    $E = E \cup \{ \langle p_i, BELONGTO, d \rangle, \langle d, INV\_BELONGTO, p_i \rangle \}$ ;
  9   foreach  $s_j \in Seeds$  do
 10   |  $E = E \cup \{ \langle s_j, FIND, p_i \rangle, \langle p_i, INV\_FIND, s_j \rangle \}$ ;
 11    $\{wrappers\} = GenerateWrappers(Seeds, \{p_i\})$ ;
 12   if  $\{wrappers\} \neq \emptyset$  then
 13   | foreach  $w_j \in \{wrappers\}$  do
 14   | |  $V = V \cup \{w_j\}$ ;
 15   | |  $E = E \cup \{ \langle p_i, DRIVE, w_j \rangle, \langle w_j, INV\_DRIVE, p_i \rangle \}$ ;
 16   | |  $\{candidates\} = Extract(w_j, p_i)$ ;
 17   | | if  $\{candidates\} \neq \emptyset$  then
 18   | | | foreach  $c_n \in \{candidates\}$  do
 19   | | | |  $V = V \cup \{c_n\}$ ;
 20   | | | |  $E = E \cup$ 
 21   | | | |  $\{ \langle w_j, EXTRACT, c_n \rangle, \langle c_n, INV\_EXTRACT, w_j \rangle \}$ ;
 22   if  $siblingFlag = true$  then
 23   |  $G' = extractingOverSiblingPages'(N_p, N, Seeds)$ , where  $G' = (V', E')$ ;
 24    $V = V \cup V', E = E \cup E'$ ;
 25 return  $G$ ;

```

value for each node in the graph. The PageRank values are then used to rank nodes of a same type, such as candidate t-uples, domains and pages. The top N_c candidate t-uples are returned as output to the users. Moreover, we compare the ranking results of the domains with that given by Google Toolbar. The intuition is that a PageRank value obtained by our scheme will determine the *relevance* of a domain for the specific query (the seed t-uples) given to the search engine. Besides, we also obtain a ranking list of pages, which indicates the degree of a page related to the seeds and the target relations. The above claims can be verified in section 5.3.

Algorithm 8: ExtractOverSiblingPages'($N_p, N, Seeds$)

Input: $N_p, N, Seeds = \{s_1, s_2, \dots, s_{N_s}\}$;
Output: $G = \{V, E\}$;

- 1 $V = \emptyset, E = \emptyset$;
- 2 $V = V \cup Seeds$;
- 3 $SeedPages = \text{FetchSeedPages}(N_p, Seeds)$;
- 4 **foreach** $\langle p_1, p_2, \dots, p_{N_s} \rangle \in SeedPages$ **do**
- 5 **foreach** $p_i \in \langle p_1, p_2, \dots, p_{N_s} \rangle$ **do**
- 6 $V = V \cup p_i$;
- 7 $E = E \cup \{ \langle s_i, FIND, p_i \rangle, \langle p_i, INV_FIND, s_i \rangle \}$;
- 8 $SiblingPages = \emptyset$;
- 9 $\{wrappers\} = \text{GenerateWrappers}(Seeds, \{p_1, p_2, \dots, p_{N_s}\})$;
- 10 **if** $\{wrappers\} \neq \emptyset$ **then**
- 11 $urlpre = \text{CommonPrefix}(p_1, p_2, \dots, p_{N_s})$;
- 12 $domain = \text{Domain}(p_1)$;
- 13 $V = V \cup \{domain\}$;
- 14 **foreach** $p_i \in \{p_1, p_2, \dots, p_{N_s}\}$ **do**
- 15 $E = E \cup \{ \langle p_i, BELONGTO, domain \rangle, \langle domain, INV_BELONGTO, p_i \rangle \}$;
- 16 **foreach** $w_j \in \{wrappers\}$ **do**
- 17 $V = V \cup \{w_j\}$;
- 18 $E = E \cup \{ \langle p_i, DERIVE, w_j \rangle, \langle w_j, INV_DERIVE, p_i \rangle \}$;
- 19 $SiblingPages = \{p_1, p_2, \dots, p_{N_p}\} = \text{Find}(urlpre \text{ site:} domain, N_p)$;
- 20 **foreach** $p_i \in SiblingPages$ **do**
- 21 $V = V \cup \{p_i\}$;
- 22 $E = E \cup \{ \langle p_i, BELONGTO, domain \rangle, \langle domain, INV_BELONGTO, p_i \rangle \}$;
- 23 **foreach** $s_j \in Seeds$ **do**
- 24 $E = E \cup \{ \langle s_j, FIND, p_i \rangle, \langle p_i, INV_FIND, s_j \rangle \}$;
- 25 **foreach** $w_i \in \{wrappers\}$ **do**
- 26 **foreach** $p_j \in SiblingPages$ **do**
- 27 $\{candidates\} = \text{Extract}(w_i, p_j)$, where $\forall c \in \{candidates\}$,
 $c = \langle attr_1, attr_2, \dots, attr_N \rangle$;
- 28 **foreach** $c_l \in \{candidates\}$ **do**
- 29 $V = V \cup \{c_l\}$;
- 30 $E = E \cup \{ \langle w_i, EXTRACT, c_l \rangle, \langle c_l, INV_EXTRACT, w_i \rangle \}$;
- 31 **return** G ;

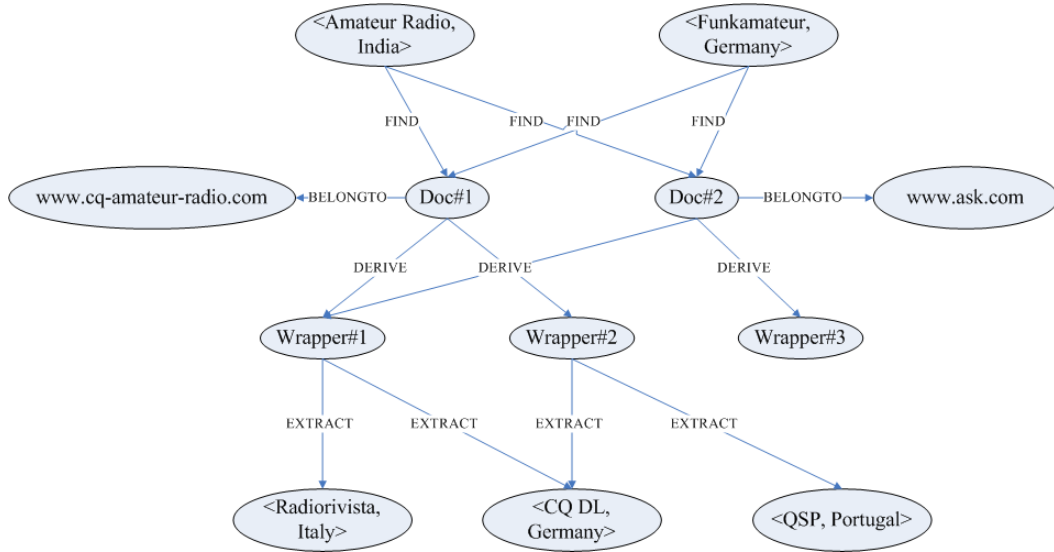


Figure 4.4: Example of part of an entity graph.

4.5 Bootstrapping of STEP

Bootstrapping is an effective iterative process in which a system uses the output of the previous iteration as input of next iteration to improve the performance, such as in literature [Brin 1998, Etzioni 2005, Talukdar 2006, Wang 2008]. We consider applying bootstrapping techniques to STEP to improving the performance. First, as can be seen, the input to STEP is a set of seed t-uples that belong to the same semantic class; the output is a ranked list of candidate t-uples that belong to the same class. To bootstrap STEP, we select the top N_s (recall N_s is the number of seed t-uples) candidate t-uples that are not used as seeds before. These top N_s t-uples are used as input to STEP to perform next iteration. This process can be run in several iterations. Algorithm 9 shows the bootstrapping process of STEP.

To be precise, given a set of amateur radio magazines as seeds, $\{ \langle \textit{Amateur Radio, India} \rangle, \langle \textit{Funkamateur, Germany} \rangle \}$, a ranking list of amateur radio magazines is returned after one iteration. The top ten candidates are illustrated in Table 4.9. Thus, according to the bootstrapping process in Algorithm 9, the seeds for the second iteration is $\{ \langle \textit{QTC Amatörradio, Sweden} \rangle, \langle \textit{RadioRivista, Italy} \rangle \}$.

Algorithm 9: Bootstrapping algorithm of STEP

Input: $Seeds = \{s_1, s_2, \dots, s_{N_s}\}, I$;
Output: $Candidates$;

- 1 $i = 0$;
- 2 $Candidates = \emptyset$;
- 3 $SeedSet = \emptyset$;
- 4 $SeedSet = SeedSet \cup Seeds$;
- 5 Using $Seeds$ as input, run STEP once to obtain a ranked list of candidates, i.e. $C = \{c_1, c_2, \dots\}$;
- 6 $Candidates = Candidates \cup C$;
- 7 Select top N_s candidates (i.e., $\{t_1, t_2, \dots, t_{N_s}\}$) that are not used as $Seeds$ before from the ranked list C , i.e. $\forall t_i \notin SeedSet$;
- 8 $Seeds = \{t_1, t_2, \dots, t_{N_s}\}$;
- 9 $i = i + 1$;
- 10 **if** $i < I$ **then**
- 11 Go to step 4;
- 12 **return** $Candidates$;

Top ID	Candidate
1	<Amateur Radio, Germany>
2	<Funkamateur, India>
3	<QTC Amatörradio, Sweden>
4	<RadioRivista, Italy>
5	<SV NEA, Greece>
6	<CQ DL, Germany>
7	<Onda corta, Mexico>
8	<Radio ZS, South Africa>
9	<Radioamatööri, Finland>
10	<Radio, Russia>

Table 4.9: Top ten candidate t-uples after one iteration.

If the seeds are chosen properly, the bootstrapping can improve the performance, as illustrated in section 5.3. However, if an *incorrect* or *spurious* candidate t-uple is chosen as a seed, it will greatly decrease the performance of next iteration. Fortunately, it does not occur in the above example, since all the top ten candidate t-uples are correct members of the semantic class of amateur radio magazines.

Performance Evaluation

Contents

5.1 Datasets	58
5.2 Evaluation Metric	61
5.3 Results	62
5.4 Discussions	74

In this chapter, we present and discuss the results of extensive experiments performed to verify the effectiveness of solutions or techniques we propose for STEP. Firstly, we describe the datasets that we manually collected from the Web as baseline for different purposes in section 5.1. Next, the evaluation metric used in this thesis is described in section 5.2. A detailed experimental results and a full analysis of these results are summarized in section 5.3, which verifies the effectiveness of our proposed solutions. Finally, in section 5.4 gives a further consideration of the experiment. Now, we start with the baseline datasets used in this thesis.

5.1 Datasets

In this thesis, we collect 15 datasets mainly from Wikipedia and other websites, and use them as the *baseline* to compare our results. Of course the Wikipedia Web pages were removed from the set of Web pages collected by STEP while extracting the desired t-uples. A brief overview of the 15 datasets is given in Table 5.1. The table indicates the *id* we assign for the dataset, the name, the arity of t-uples in the

ID	Name	Arity	Language	Size	Seeds
D1	Amateur Radio	2	English	50	{<Amateur Radio,India>, <CQ-PA,Netherlands>}
D2	Countries Death Rate	2	English	196	{<Liberia,18.3>, <Chad,15.4>}
D3	Federation Abbreviation	2	English	711	{<ARB,Administrative Review Board>, <VOA,Voice of America>}
D4	Federation Units	2	English	26	{<Austria,States of Austria>, <Iraq,Governorates of Iraq>}
D5	FIFA Codes	2	English	208	{<Algeria,ALG>, <Andorra,AND>}
D6	NBA Teams	2	Chinese English	30	{<新奥尔良黄蜂,New Orleans Hornets>, <芝加哥公牛,Chicago Bulls>}
D7	Ship IMO	2	English	150	{<Aeolian Vision,9483554>, <Aegean Star,7502942>}
D8	SoC Courses	2	English	147	{<CG1101,Programming Methodology>, <CP3201,Industry Seminar>}
D9	Taiwan Mayors	2	Chinese	25	{<桃园县,朱立伦>, <金门县,李炆烽>}
D10	US Governors	2	English	56	{<John Baldacci,Maine>, <Arnold Schwarzenegger, California>}
D11	Wiki Focus	2	English	88	{<NotePub,General purpose>, <TermWiki, Terminology management platform>}
D12	Countries Imports	3	English	450	{<Germany, (931,000,000,000),2009>, <Russia, (302,000,000,000),2008>}
D13	Singapore Films	3	English	106	{<Lucky Number,Lam Po Ko, DS Movie Production>, <Sharp Pencil,Gallen Mei, Under Pressure Pictures>}
D14	Countries Population	4	English	225	{<Czech Republic,(10,535,811), (March 31, 2011),0.15%>, <Ethiopia,(82,101,998), 2011,1.18%>}
D15	US presidents	4	English	44	{<George Washington, (April 30, 1789), (March 4, 1797),No party>, <Thomas Jefferson, (March 4, 1801) , (March 4, 1809),Democratic>}

Table 5.1: Baseline datasets used in the performance evaluation.

dataset, the language, the size of the dataset and the seeds used in the experiments. A complete description of the data sets with sample results is given in appendices A.1 to A.15.

As can be seen, these datasets involve a wide range of subjects, such as sports (e.g., D5, D6), economics (e.g., D12), entertainment (e.g., D13), politics (e.g., D15), and so on. These different datasets are chosen to verify the effectiveness of STEP in various practical scenarios. In particular, these datasets contain t-uples with different arities, i.e. datasets from D1 to D11 are with arity 2, D12 and D13 with arity 3, and datasets D14 and D15 with arity 4. In some of our experiments we used bilingual dataset, i.e. Chinese and English t-uples in dataset D6.

In most of our experiments we chose 2 seeds for the above datasets. Note that, having fewer seeds (as compared to a large number of seeds) is a more stringent test for STEP. This is partially because more instances of *expected* t-uples STEP can use (for constructing wrappers), it is likely to fetch candidate t-uples more accurately. To verify this intuition, we performed another set of experiments in which we used more than 2 seeds. To be precise, we use datasets D1 (arity 2), D13 (arity 3) and D15 (arity 4) for experiments in which we vary the number of input seeds from 2 to 10.

We are also interested in the impact of verifying arity of seed t-uples on the quality of results. Because of the significant difference in the source quality and nature of different datasets, the impact of arity on the results cannot be justified by comparing among different datasets. To resolve this issue, we performed experiments in which we fix datasets and increase the arity of the seed t-uples. To be precise, we used datasets D14 and D15 in which we could vary arity from 2 to 4.

The effectiveness of constructing wrappers to extract t-uples from sibling pages is examined over datasets D7 and D8, since the candidate t-uples of the two datasets mostly distribute on sibling pages of a website.

Besides, we also study the impact of introducing *domains* into the graph over

D7, which one can find sources from several different websites (i.e., *domains*).

Moreover, we also attempt to study the impact of number of Web pages on the final result. Thus, a comparison of performance is summarized over datasets D5, D13 and D15 with different number of Web pages returned a search engine.

As mentioned in the introduction chapter, the nature of seeds can be also a vital factor that impacts the performance. Hence, we perform experiments over dataset D1 with different choices of seeds to verify our above claim explicitly.

Furthermore, an experiment is performed over datasets D1, D13 and D15 to study the impact of bootstrapping techniques on our STEP.

Finally, using dataset D1, we illustrate a byproduct of our graph based ranking mechanism, a ranking list of documents.

5.2 Evaluation Metric

For the sake of evaluating performance in different aspects more explicitly in section 5.3, we use *precision* and *recall* separately instead of the commonly used *MAP*, which is a measure combining them together. Another ranking metric called Mean Reciprocal Rank (i.e., MRR) is also not appropriate, since it only evaluates the correctness of the seeds. Let R and B be the ranked lists returned by STEP and *baseline*, respectively. The *precision* (i.e., p) and *recall* (i.e., r) are defined as follows.

$$p = \frac{\sum_{i=1}^{|R|} \text{NewEntity}(i)}{|R|}; r = \frac{\sum_{i=1}^{|R|} \text{NewEntity}(i)}{|B|} \quad (5.1)$$

$\text{NewEntity}(i)$ in Eq.(5.1) is a binary function [Wang 2009], which returns 1 if a) the extracted t-uple at i matches any true relation, and b) there exist no other extracted t-uples at rank less than i that is of the same relation as the one at r . It returns 0 otherwise.

Parameter	Value
N	2, 3, 4
N_c	10, 20, 50, 100, 200
N_p	10, 20, 50, 100
N_s	2, 4, 6, 8, 10
<i>siblingFlag</i>	true, false

Table 5.2: Parameter setting.

5.3 Results

In this section, we will report and analyze all the results we obtain in the experiments mentioned above. Note that "n/a" in tables represents the non-availability of the results. To be precise, when SEAL or STEP does not extract enough candidates according to varying N_c , in particular when N_c is 100 or 200, then results cannot be computed. Parameter settings in the experiments are illustrated in Table 5.2.

Comparison between DIPRE and STEP. In this section, we compare the performance of DIPRE [Brin 1998] with STEP. DIPRE exploits the duality between patterns and relations to expand a small given set. It is evaluated by extracting pairs of $\langle author, book-title \rangle$ from a large collection of Web pages. Given five pairs of authors and book-titles, DIPRE extracts over 15,000 unique books over several iterations in which 19 out of 20 randomly-sampled pairs are found with correct book-titles. However, we take the top two pairs, i.e. $\{\langle Isaac Asimov, The Robots of Dawn \rangle, \langle David Brin, Startide Rising \rangle\}$ as seeds for STEP. After one iteration, it extracts 1,223 pairs of books from the top 100 pages returned by Google. Thereafter, we randomly choose 20, 30, 50, 100 pairs from the 1,223 t-uples. And then we manually check whether each pair among the randomly chosen set is a real book-title and its corresponding author. In other words, we define *accuracy* (i.e., α) of the randomly chosen set θ as follows.

$$\alpha = \frac{\sum_{i=1}^{|\theta|} RealPair(i)}{|\theta|}; \quad (5.2)$$

$RealPair(i)$ in Eq.(5.2) is a binary function, which returns 1 if a) the second attribute of the extracted t-uple at i is a real book-title, b) the first attribute of the extracted t-uple at i is the corresponding author of this book, and c) there exist no other extracted t-uples at rank less than i that is of the same book as the one at i . It returns 0 otherwise.

Approach	20	30	50	100
DIPRE	0.95	n/a	n/a	n/a
STEP	1	1	1	0.96

Table 5.3: Comparison of accuracy of DIPRE and STEP with varying size of randomly choosing set ($|\theta| = 20, 30, 50, 100$).

Results in Table 5.3 compares the accuracy of DIPRE with that of STEP while ranging the size (i.e., θ) of the randomly chosen set from 20, 30, 50 to 100. Our work is different with DIPRE in the fact that the latter not only requires the wrappers to bracket all occurrences of seeds, but also requires the wrappers to be exactly matched while extracting candidates. Hence, we may claim the way that DIPRE constructs wrappers is too rigid. Since it is difficult to perform a fair comparison between DIPRE and STEP, we focus on the comparison between SEAL and STEP in the next sections with two more fair and precise metrics, i.e. *precision* and *recall*.

Comparison between SEAL and STEP. In this experiment, we compare the performance of SEAL with STEP. In order to compare with STEP, we also extend SEAL to work on t-uples with arity greater than two. We refer to this version of SEAL as SEAL+ in the rest of the thesis. To simplify, we just consider extracting t-uples having attributes in the same order as the seeds. We present the precision (i.e., p) and recall (i.e., r) of top N_c (N_c varying from 10 to 100) candidates of SEAL(+) and STEP in Table 5.4 and 5.5, respectively. It can be seen from Table 5.4 that STEP outperforms SEAL significantly, especially in datasets D1, D7, and D13. For the three datasets, there are no exact matching middle contexts among the adjacent attributes of all the seeds. Therefore, the EM approach fails to

Data	Scheme	10	20	50	100
D1	SEAL	n/a	n/a	n/a	n/a
	STEP	0.80	0.85	0.72	0.46
D2	SEAL	0.40	0.30	0.36	0.36
	STEP	0.40	0.35	0.42	0.43
D3	SEAL	1	1	1	0.82
	STEP	1	1	1	0.99
D4	SEAL	0.90	0.45	0.32	0.22
	STEP	0.90	0.45	0.42	0.23
D5	SEAL	1	0.85	0.90	0.85
	STEP	1	0.90	0.96	0.92
D6	SEAL	1	0.85	n/a	n/a
	STEP	1	0.95	0.60	0.30
D7	SEAL	n/a	n/a	n/a	n/a
	STEP	1	1	1	n/a
D8	SEAL	0.80	0.67	n/a	n/a
	STEP	0.90	0.95	0.98	0.99
D9	SEAL	1	1	0.46	n/a
	STEP	1	1	0.46	n/a
D10	SEAL	1	0.85	0.56	0.39
	STEP	1	0.95	0.58	0.45
D11	SEAL	1	1	0.82	0.41
	STEP	1	1	0.82	0.41
D12	SEAL+	1	1	1	1
	STEP	1	1	1	1
D13	SEAL+	n/a	n/a	n/a	n/a
	STEP	0.90	0.85	0.64	0.56
D14	SEAL+	1	1	0.84	0.61
	STEP	1	1	0.86	0.61
D15	SEAL+	1	1	0.84	0.42
	STEP	1	0.95	0.84	0.42

Table 5.4: Comparison of precision of top N_c ($N_c = 10, 20, 50, 100$) candidates returned by SEAL and STEP).

Data	Scheme	10	20	50	100
D1	SEAL	n/a	n/a	n/a	n/a
	STEP	0.16	0.33	0.71	0.90
D2	SEAL	0.02	0.03	0.09	0.18
	STEP	0.02	0.04	0.11	0.22
D3	SEAL	0.01	0.03	0.07	0.12
	STEP	0.01	0.03	0.07	0.14
D4	SEAL	0.35	0.35	0.62	0.85
	STEP	0.35	0.35	0.81	0.89
D5	SEAL	0.05	0.08	0.22	0.41
	STEP	0.05	0.09	0.23	0.44
D6	SEAL	0.33	0.57	0.90	n/a
	STEP	0.33	0.63	1	1
D7	SEAL	n/a	n/a	n/a	n/a
	STEP	0.07	0.13	0.33	n/a
D8	SEAL	0.05	0.07	n/a	n/a
	STEP	0.06	0.13	0.33	0.67
D9	SEAL	0.40	0.80	0.92	n/a
	STEP	0.40	0.80	0.92	n/a
D10	SEAL	0.18	0.30	0.50	0.70
	STEP	0.18	0.34	0.52	0.80
D11	SEAL	0.11	0.23	0.47	0.47
	STEP	0.11	0.23	0.47	0.47
D12	SEAL+	0.02	0.04	0.11	0.22
	STEP	0.02	0.04	0.11	0.22
D13	SEAL+	n/a	n/a	n/a	n/a
	STEP	0.09	0.16	0.30	0.53
D14	SEAL+	0.04	0.09	0.19	0.27
	STEP	0.04	0.09	0.19	0.27
D15	SEAL+	0.23	0.46	0.96	0.96
	STEP	0.23	0.43	0.96	0.96

Table 5.5: Comparison of recall of top N_c ($N_c = 10, 20, 50, 100$) candidates returned by SEAL and STEP).

construct any wrappers, and hence SEAL fails to extract any candidate t-uples. On the other hand, the RE approach allows the wrappers to be more flexible to extract candidate t-uples. This result highlights the fact that RE based wrappers present a viable solution for extending atomic set expansion to set of t-uples expansion. In particular, when N_c is set at 10, STEP’s precision is 0.8, 1.0, and 0.9, respectively, from D1, D7, and D13 datasets. When N_c increases beyond 10, STEP’s performance remains better than SEAL’s performance.

In some situations, however, the flexibility of RE wrappers may contribute to a

minor decay in the performance. For example, note the result of STEP in D15 when N_c is set at 20 as shown in Table 5.4. There is a lower precision for STEP (0.95) as compared to SEAL (1). The reason for this behavior is that RE based wrappers are too flexible and they tend to extract some *synonymous* or *junk* candidates, which reduces the precision. Overall RE (STEP) performs better than EM (SEAL).

Impact of varying the number of seed t-uples. The number of seed t-uples may also impact the performance. We can expect that when the number of seed t-uples is increased, the number of pages (fetched by search engines) that contain all the seeds may also increase. Moreover, the wrappers are constructed more precisely. Overall, that may result in increased precision. On the other hand, it may enforce stricter constraints which require more seeds to occur in similar contexts on the pages. Thus it may decrease the recall significantly. To verify the above claim, we perform experiments in which we vary the number of seed t-uples from 2, 4, 6 to 10 while using datasets with different arities, i.e, D1 (arity=2), D13 (arity=3) and D15 (arity=4).

Table 5.6 illustrates the precision and recall at the top 20 candidate t-uples while varying the number of seed t-uples from 2, 4, 6 to 10. As the number of seed t-uples increases from 2 to 4, the precision of the top 20 candidate t-uples increases, e.g. over D1, D13, and D15, the precision increase by 12%, 12% and 5%, respectively. However, when the number of seed t-uples is increased to 8 or 10, STEP either fails to fetch any Web pages that contain all the seed t-uples or no wrappers are constructed to extract the candidate t-uples. Overall, it is consistent with our expectation.

Impact of varying the arity of seed t-uples. Another factor that may influence performance is the arity of the seed t-uples. It is very similar to the impact of number of seed t-uples on precision and/or recall of the result. Specifically, we may expect that while increasing the arity of seed t-uples, on one hand, it is more likely for a search engine to return more relevant Web pages that contain more information of each seed t-uple. Therefore, it may filter such Web pages that can produce fuzzy

Dataset		2	4	6	8	10
D1	<i>precision</i>	0.85	0.95	1	n/a	n/a
	<i>recall</i>	0.33	0.37	0.39	n/a	n/a
D13	<i>precision</i>	0.85	0.95	1	1	1
	<i>recall</i>	0.16	0.18	0.19	0.19	0.19
D15	<i>precision</i>	0.95	1	1	1	n/a
	<i>recall</i>	0.43	0.45	0.45	0.45	n/a

Table 5.6: Comparison of precision and recall of top 20 candidates with varying number of seeds ($N_s = 2, 4, 6, 8, 10$).

wrappers which in turn extract lots of *junks*. In such a way, it potentially improves the precision. To verify this statement, we perform an experiment in which we vary the arity of seed t-uples as 2, 3 and 4 over datasets D14 and D15. Table 5.7 illustrates the precision and recall at the top 20 candidate t-uples while varying the arity of seed t-uples as 2, 3 and 4. As the arity of seed t-uples increases from 2 to 3, the precision of the top 20 candidate t-uples increases greatly. Specifically, over D14, D15, the precision increases by 33% and 6%, respectively. Hence, it is in accord with our claim. On the other hand, we can imagine that while increasing the arity of seed t-uples, it requires more information of each seed t-uple to appear on a single page, which can be even harder to be satisfied in reality. It is our future work to extend our system to work on scenarios where attributes of a seed t-uple scatter on several Web pages.

Dataset		2	3	4
D14	<i>precision</i>	0.75	1	1
	<i>recall</i>	0.07	0.09	0.09
D15	<i>precision</i>	0.90	0.95	0.95
	<i>recall</i>	0.36	0.43	0.43

Table 5.7: Comparison of precision and recall of top 20 candidates with varying arity of seeds and target relations ($N = 2, 3, 4$).

Impact of extracting t-uples over sibling pages. In this section, we illustrate the effectiveness of STEP in terms of extracting t-uples from sibling Web pages

while using D7 and D8 datasets. In Table 5.8 and 5.9, we present the precision and recall of top N_c ($N_c=10, 20, 50, 100, 200$) candidates returned by STEP with and without extracting t-uples over sibling pages. We did not notice much change for the rest of the datasets, e.g. no siblings Web pages that contain candidate t-uples were discovered by STEP. Therefore, there is hardly any change in the precision or recall values for the rest of the datasets.

Dataset	Sibling Pages	10	20	50	100	200
D7	Without	1	1	1	n/a	n/a
	With	1	1	1	1	1
D8	Without	0.90	0.95	0.98	0.99	0.67
	With	0.90	0.95	0.98	0.99	0.70

Table 5.8: Comparison of precision of top N_c ($N_c = 10, 20, 50, 100, 200$) candidates with and without extraction over sibling pages.

Dataset	Sibling Pages	10	20	50	100	200
D7	Without	0.07	0.13	0.33	n/a	n/a
	With	0.07	0.13	0.33	0.67	0.71
D8	Without	0.06	0.13	0.33	0.67	0.91
	With	0.06	0.13	0.33	0.67	0.95

Table 5.9: Comparison of recall of top N_c ($N_c = 10, 20, 50, 100, 200$) candidates with and without extraction over sibling pages.

By considering extraction over the sibling Web pages, we improve the precision of the ranked list. For example, as shown in Table 5.8, STEP is able to increase the precision from 0.67 to 0.70 for top 200 candidates (i.e., when N_c is 200) over D8. In some cases, STEP can find a desired number of t-uples with extraction from the sibling pages, which otherwise were not available before. Because STEP finds more candidate t-uples over sibling Web pages, it actually improves the recall as well. More specifically, as can be seen from the results for D7 in Table 5.9, with extraction over sibling Web pages, the final recall (0.71) of STEP for top 200 candidate t-uples is over two times of that (0.33) of STEP without extraction over sibling Web pages.

Impact of introducing domains. In Table 5.10 we compare the ranking of

Domain	STEP Rank(value)	Google Toolbar Rank(value)
www.vesseltracker.com	1 (0.00862)	2 (5/10)
seaagent.com	2 (0.00732)	4 (3/10)
www.shipspotting.com	3 (0.00632)	3 (4/10)
www.ship.gr	4 (0.00197)	3 (4/10)
surfpack.com	4 (0.00197)	5 (2/10)
twitter.com	4 (0.00197)	1 (9/10)

Table 5.10: Comparison of domain ranking of STEP and Google Toolbar on D7.

domains returned by STEP and Google Toolbar over D7. The Google Toolbar's PageRank¹ indicates the popularity of a visited page as an integer number between 0 and 10. A PageRank of 10 reflects the most popular. The least popular page will have a PageRank of 0. These comparisons can be interpreted as following. STEP's ranking of domains reflects the relevance of the domain with respect to the semantic class that we intend to expand. While Google Toolbar indicates the importance or popularity of the respective domain over the whole Web, STEP's ranking list is useful in the sense that it tells the user the importance or relevance of the domains for the particular query or the semantic set being expanded. For example, note the ranking of websites "www.vesseltracker.com" and "twitter.com" in Table 5.10. The vesseltracker site is more popular to the users of the query (ships) than the twitter site, which is more popular in general. Intuitively, such cases suggest that the ranking list of domains given by STEP is more interesting and meaningful than that given by Google Toolbar.

Impact of number of Web pages. The number of Web pages, i.e. N_p , returned by a search engine also has great impact on the performance. As the number of Web pages related to the seeds increases, it is likely to extract more candidate t-uples. Consequently, it will increase the performance, both *precision* and *recall*. To verify this claim, we vary the number of Web pages, i.e. N_p , from 10, 20, 50, and 100. Without loss of generality, we perform comparison of performance

¹http://en.wikipedia.org/wiki/PageRank#Google_Toolbar

over datasets with different arities, i.e. D5 (arity=2), D13 (arity=3), and D15 (arity=4).

Dataset	10	20	50	100
D5	0	0.18	0.56	0.92
D13	0	0	0.47	0.56
D15	0	0	0	0.42

Table 5.11: Comparison of precision of top 100 candidates with varying number of Web pages ($N_p = 10, 20, 50, 100$).

Dataset	10	20	50	100
D5	0	0.09	0.27	0.44
D13	0	0	0.44	0.53
D15	0	0	0	0.96

Table 5.12: Comparison of recall of top 100 candidates with varying number of Web pages ($N_p = 10, 20, 50, 100$).

Table 5.11 illustrates the comparison of *precision* of the top 100 (i.e., $N_c = 100$) candidates over datasets D5, D13, and D15 while varying the number of relevant Web pages, i.e. N_p . As can be seen from this figure, when the number of relevant Web pages is increased, the precision of all the three dataset increases significantly. For example, when N_p is increased from 50 to 100, the precision of the top 100 candidates over datasets D5, D13 increases by 64% and 19%, respectively. Besides, by increasing N_p , more candidates can be discovered. That means, increasing N_p will also increase the recall. Taking D13 in Table 5.12 for instance, STEP only extracts about 50 candidates from the top 20 web pages related to seeds $\{<Lucky Number, Lam Po Ko, DS Movie Production>, <Sharp Pencil, Gallen Mei, Under Pressure Pictures>\}$. However, more than 100 candidates can be extracted from the top 50 Web pages. Hence, the experiment validates the above statement.

Impact of choosing seeds. The choice of seeds can also affect the performance greatly. The reason is apparent. A good choice of seeds means that the context of occurrences of seeds is representative for that of most other candidates of the target semantic class on the pages. A good choice of seeds will increase the performance; however, if the choice of seeds is not representative, the performance will drop dramatically. In the worst case, the surrounding contexts of seeds may be completely different. Hence, there will be no wrappers that can be constructed, which will result in no candidates.

To illustrate this phenomenon clearly, we perform an experiment over dataset

Seeds	10	20	50	100
{<Amateur Radio,India>, <Break In,New Zealand>}	1	n/a	n/a	n/a
{<Amateur Radio,India>, <CQ-PA,Netherlands>}	0.8	0.85	0.72	0.46
{<Amateur Radio,India>, <CQ-QSO,Belgium>}	1	0.95	0.98	n/a

Table 5.13: Comparison of precision of top N_c ($N_c=10, 20, 50, 100$) candidates with different choices of seeds.

<i>suffix</i>	</td> </tr> <tr> <td><i>
<i>middle</i> ₁	(</i></td> <td>
<i>prefix</i>	</td> <td>English</td> <td>

Table 5.14: Another example of wrapper. Candidate t-uples occurring in the form of "*suffix*[Magazine Name]*middle*₁[Country]*prefix*" are extracted by this wrapper from the page shown in Table 4.3.

D1 by using different seeds. The comparison of precision of top N_c ($N_c=10, 20, 50, 100$) candidates using different seeds is shown in Table 5.13. In this case, if {<Amateur Radio, India>, <Break In, New Zealand>} is used as seeds, although their context are similar and wrappers can be constructed, no candidates will be generated. Because their contexts are too similar, the wrappers constructed are too stringent. Thus, fewer candidates will be generated. For instance, if {<Amateur Radio, India>, <Break In, New Zealand>} is used as seeds, one wrapper constructed on the page illustrated in Table 4.3 is shown in Table 5.14. In this wrapper, it requires the prefix of middle context between the name of magazine and its country of origin to be end with digitals followed by a slash followed by digitals. As can be seen, there are no more t-uples that are matched on the partial page in Table 4.3.

On the Contrast, if seeds are chosen like {<Amateur Radio, India>, <CQ-PA, Netherlands>}, the wrappers constructed can be too flexible. They will extract not

only correct candidates but also *junks*. Consequently, it will also decrease the performance. In this example, we can claim that $\{\langle \textit{Amateur Radio, India} \rangle, \langle \textit{CQ-QSO, Belgium} \rangle\}$ is a good choice of seeds. Over all, it can be inferred that carefully choosing seeds will obtain elegant performance. However, it is non-trivial to determine how to choose a good set of seeds. Perhaps, the bootstrapping technique introduced in the following can be helpful for this situation in some way.

Impact of bootstrapping. Bootstrapping is an effective iterative process in which a system uses the output of the previous iteration as input to improve the performance, such as in literature [Brin 1998, Etzioni 2005, Talukdar 2006, Wang 2008]. All the experimental results above are obtained through one iteration run. We consider applying bootstrapping techniques to STEP to improving the performance.

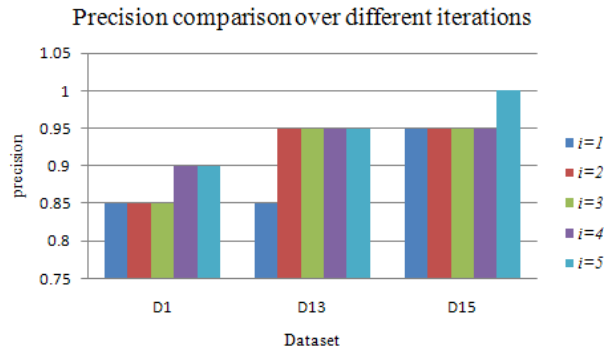


Figure 5.1: Comparison of precision of top 20 candidates in different iterations ($i = 1, 2, 3, 4, 5$).

In this experiment, we set the number of seed t-uples and the number of iterations to be 2 and 5, i.e. setting $N_s = 2, I = 5$ in Algorithm 9. Without loss of generality, we perform the experiment over datasets with different arities, i.e. D1 (arity=2), D13 (arity=3), and D15(airty=4). We compare both *precision* and *recall* of the top 20 (i.e., $N_c = 20$) candidates over D1, D13, and D15 from iteration 1 to 5 in a bootstrapping process in Figure 5.1 and Figure 5.2, respectively. As can be seen from Figure 5.1, the precision of top 20 candidates increases as more iterations are run, e.g. the precision of top 20 candidates over D13 increases by 12% through one

extra iterations compared to that of the first iteration. Consequently, the recall of top 20 candidates also increases while performing more iterations, which can be shown in Figure 5.2.

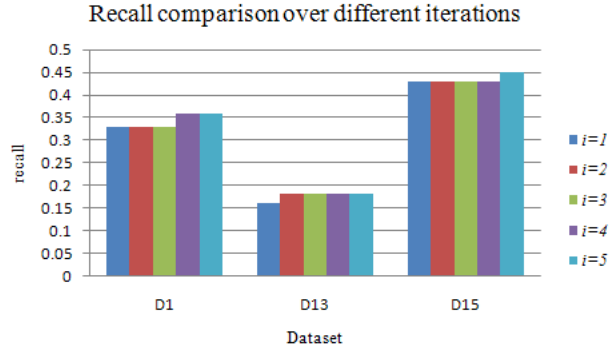


Figure 5.2: Comparison of recall of top 20 candidates in different iterations ($i = 1, 2, 3, 4, 5$).

A byproduct: ranking of Web pages. Since we build a graph which integrates all the entities and relations occurring in the extraction process, a run of ranking method will also produce a ranked list of other entities except for the candidate t-uples. One byproduct of interest is a ranking list of Web pages. It is interesting because the ranking of the Web pages indicates which pages are more *relevant* to the given seeds and the target relations to be extracted.

Table 5.15 illustrate the top ten Web pages over D1, given the seeds as $\{ \langle \textit{Amateur Radio}, \textit{India} \rangle, \langle \textit{CQ-PA}, \textit{Netherlands} \rangle \}$. The top sixth Web page is "www.ask.com/wiki/List_of_amateur_radio_magazines". It is said that this page is more relevant to the two seed amateur radio magazines and the semantic class of "*Amateur Radio Magazines*" than other pages below. It makes certain sense. Since as can be seen from the URL, this page summarizes a list of amateur radio magazines, which is essentially the target relation that we want to expand. Compared with the top eleventh URL, "www.eqsl.cc/qsldata/CountryList.cfm?Country=NETHERLANDS", it illustrates a list of users of some product (i.e., electronic QSL card) from Netherlands. Al-

Top ID	PageRank Value	URL
1	0.0374	www.mshtawy.com/en-wiki.php? title=List_of_amateur_radio_magazines
1	0.0374	wikiand.com/wiki/ List_of_amateur_radio_magazines
1	0.0374	pediaview.com/openpedia/ List_of_amateur_radio_magazines
1	0.0374	www.territorioscuola.com/wikipedia/en.wikipedia.php? title=List_of_amateur_radio_magazines
1	0.0374	www.secret-bases.co.uk/wiki.php? url=wiki/List_of_amateur_radio_magazines
6	0.0362	www.rescue.kate-jenter.com/ p-List_of_amateur_radio_magazines
6	0.0362	www.house.giftedamersexdating.com/ p-List_of_amateur_radio_magazines
6	0.0362	www.ask.com/wiki/ List_of_amateur_radio_magazines
6	0.0362	uk.ask.com/wiki/ List_of_amateur_radio_magazines
10	0.0356	abitabout.com/ List+of+amateur+radio+magazines

Table 5.15: Top ten Web pages ranked by PageRank.

though it involve an attribute (i.e., *Netherlands*) of the given seeds, this URL is certainly not relevant to the semantic class of the seeds.

Besides, it is noted that this ranking of Web pages is not necessarily equivalent to the ranking by the number of candidate t-uples extracted on these pages. To compare, we also rank the Web pages according to the number of candidate t-uples extracted on these pages. Using the same seeds, Table 5.16 illustrate the top ten Web pages over D1, which are ranked by the number of candidate t-uples extracted, i.e. *frequency*. For instance, the top tenth URL in Table 5.16 indicates that over 50 candidate t-uples are extracted from this page. However, the ranking of this page is ranked as the last URL while ranking by PageRank value, because most of the 50 candidate t-uples are *spurious* amateur radio magazines.

In the Appendix A, we illustrate descriptions and experimental results of each dataset used in this thesis, including the top 20 candidate t-uples, top ten domains, and top ten Web pages returned by our STEP.

Top ID	Frequency	URL
1	109	www.rescue.kate-jenter.com/ p-List_of_amateur_radio_magazines
2	107	www.house.giftedamersexdating.com/ p-List_of_amateur_radio_magazines
3	101	www.ask.com/wiki/ List_of_amateur_radio_magazines
4	98	pediaview.com/openpedia/ List_of_amateur_radio_magazines
5	97	www.territorioscuola.com/wikipedia/ en.wikipedia.php?title=List_of_amateur_radio_magazines
5	97	www.mshtawy.com/ en-wiki.php?title=List_of_amateur_radio_magazines
5	97	abitabout.com/ List+of+amateur+radio+magazines
8	92	uk.ask.com/wiki/ List_of_amateur_radio_magazines
9	91	www.secret-bases.co.uk/ wiki.php?url=wiki/List_of_amateur_radio_magazines
10	51	quick-ip-lookup.info/249.169.3/index.jsp

Table 5.16: Top ten Web pages ranked by frequency.

5.4 Discussions

It is worth noting that the order of attributes in the seed t-uples will affect the extraction of candidate t-uples. In particular, if the order of the attributes in the seed t-uples differs, or it is different from the order of the attributes on a Web page, then STEP will fail to construct a wrapper from that page. In other words, STEP will not extract any candidate t-uple from that Web page, irrespective of the fact that such a t-uple may exist on that particular Web page. Unfortunately, users may provide seed t-uples in an arbitrary order, which may affect the performance of STEP. To solve this problem, we chose the following strategy. We generate the permutations of all the attributes of each seed. Thereafter, each possible combination of every permutation of the attributes of each seed is used to construct a wrapper to extract candidate t-uples. It is a simple and comprehensive technique that extracts all possible candidate t-uples irrespective of any order of the attributes in the seeds. Unfortunately, it is computationally expensive. To be precise, if N_s is the number of

seed t-uples, then the complexity of generating *all* wrappers is $O((N!)^{N_s})$. (Recall N is the arity of the seed t-uples.) In our future work, we intend to improve the efficiency of this technique through approximation solutions.

Conclusion and Future Work

Contents

6.1 Conclusion	76
6.2 Future Work	78

In this chapter, we conclude the whole thesis to remind the reader of our contributions. Besides, we present some plans for the future work.

6.1 Conclusion

The World Wide Web is a vast and valuable repository. It is useful to extract information of interest from the Web. However, it is never a trivial task because the Web is largely unstructured and highly distributed. Extensive work has been done on this problem under various names and forms, among which set expansion is a particular technique we concern in this thesis. Set expansion is the task of finding members of a semantic class, the set, given a small subset of its members, the seeds. It is an important technique for information retrieval and data mining tasks. Many solutions proposed in the literature are restricted to expanding a unary or binary set only. In this thesis, we address a more generalized problem, expanding a set of t -uples using the Web.

To start with, we offer a taxonomy of existing set expansion systems based on several metrics, such as data source (e.g., corpus-based or Web-based), pattern construction (e.g., distributional similarity, positive and unlabeled examples learning

and wrapper induction), and arity of the seeds and target relations. Besides, the advantages and shortcomings of each category are also summarized. Through this taxonomy, we aim to give a full picture of the research context of this topic. Despite of these differences, it is observed that most of set expansion systems fall into a three-step framework, i.e. fetching relevant documents, constructing patterns and extracting candidates, and ranking candidates.

Next, we describe some background knowledge before introducing our approach, i.e. DIPRE and SEAL. They are two well-known Web-based set expansion systems, which both induce wrappers to extract unary or binary relations. However, since the way that they construct wrappers are too stringent, they cannot be properly used in high order relation extraction.

Hence, we propose a set of t-uples expansion system, STEP, which aims at generalizing set of atomic values or binary relations expansion to set of n-ary t-uples expansion. The generalization from sets of atomic values to set of t-uples raises problems at every stage of the expansion process, mainly, location of the sources, wrapper construction and extraction of candidates, and ranking of candidates. We showed that set of t-uples expansion can be achieved effectively by: (i) proposing a regular expression based approach to making the wrappers more flexible and (ii) extracting t-uples from sibling pages. We also proposed a ranking scheme, which reveals useful insights about the domains. We also integrate our STEP into a bootstrapping process to improve the performance. Besides, a byproduct of our system, a ranking list of documents, also illustrates the effectiveness of our graph based ranking mechanism.

In the experiment part, we evaluated STEP extensively and results show that it is effective in various scenarios. Besides, we also study different factors that can affect the performance and offer some constructive suggestions.

6.2 Future Work

In the course of the design, implementation and evaluation of STEP, we have identified some limitations and shortcomings of the current proposal. Future work can tackle the following issues. In section 4.2.1, we simply use a concatenation of all the seeds as keywords to fetch relevant documents. A quick check shows that different ways to make queries indeed affect the ranking of pages returned by search engines, which will in turn impact the resultant performance. In the future, we plan to discover an effective way to construct queries in order to get better performance.

Another limitation of our STEP lies in the fact that it can only extract candidate t-uples whose attributes are in the same order with that of the seeds. This limitation will greatly decrease the recall or coverage of our result. A naive way is as follows. We first generate all potential orders of the attributes in the seeds. Afterwards, for each potential order, we run our STEP once to extract candidate t-uples in the same order. However, this naive approach is significantly time-consuming because the complexity is exponential of the number of attributes in the seeds. Hence, we plan to develop an efficient approach to extract t-uples whose attributes are in arbitrary order in the future.

As shown in the experiment section, our graph based ranking mechanism is very effective and of great interest. In this thesis, the entity graph consists of five different types of nodes and eight different types of relations among these nodes as summarized in Table 4.8. In the future, we intend to include more nodes and/or relations to improve the final ranking.

Besides, we also intend to develop a set of t-uples expansion system over free text collections. A feasible idea is to factorize the high order relation into a set of lower order relations as the idea proposed in [McDonald 2005]. Thereafter, we extract instances of these lower order relations. Finally, the instances of lower order relations are reconstructed into instances of high order relations. In the future, we plan to develop a system to realize this idea.

Bibliography

- [Agichtein 2000] Eugene Agichtein and Luis Gravano. *SNOWBALL: Extracting relations from large plain-text collections*. In Proc. of the ACM Conf. on Digital Libraries, pages 85–94, 2000. (Cited on pages [12](#), [15](#) and [16](#).)
- [Badica 2004] Costin Badica and Amelia Badica. *Rule learning for feature values extraction from HTML product information sheets*. In RuleML, pages 37–48, 2004. (Cited on pages [12](#) and [13](#).)
- [Badica 2005] Costin Badica, Amelia Badica and Elvira Popescu. *Tuples extraction from HTML using logic wrappers and inductive logic programming*. In Proc. of AWIC, pages 44–50, 2005. (Cited on pages [2](#), [13](#), [15](#), [16](#) and [45](#).)
- [Brin 1998] Sergey Brin. *Extracting patterns and relations from the World Wide Web*. In Selected papers from the Int. Workshop on The World Wide Web and Databases, pages 172–183, 1998. (Cited on pages [vi](#), [ix](#), [2](#), [6](#), [9](#), [12](#), [13](#), [15](#), [16](#), [19](#), [20](#), [21](#), [22](#), [23](#), [24](#), [40](#), [55](#), [62](#) and [71](#).)
- [Cafarella 2008] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu and Yang Zhang. *WebTables: exploring the power of tables on the web*. Proc. of VLDB Endow., pages 538–549, 2008. (Cited on pages [8](#), [16](#) and [18](#).)
- [Crescenzi 2001] Valter Crescenzi, Giansalvatore Mecca, Paolo Merialdo, Università Roma, Tre Università, Basilicata Università and Roma Tre. *RoadRunner: Towards Automatic Data Extraction from Large Web Sites*. In VLDB, pages 109–118, 2001. (Cited on pages [12](#), [13](#), [15](#), [16](#) and [45](#).)
- [Elmeleegy 2009] Hazem Elmeleegy, Jayant Madhavan and Alon Halevy. *Harvesting relational tables from lists on the web*. Proc. of VLDB Endow., pages 1078–1089, 2009. (Cited on pages [8](#), [16](#) and [18](#).)

- [Etzioni 2004] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld and Alexander Yates. *Web-scale information extraction in knowitall: (preliminary results)*. In Proc. of the Int. Conf. on World Wide Web, pages 100–110, 2004. (Cited on page 14.)
- [Etzioni 2005] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld and Alexander Yates. *Unsupervised named-entity extraction from the web: an experimental study*. Artif. Intell., pages 91–134, 2005. (Cited on pages 55 and 71.)
- [Etzioni 2008] Oren Etzioni, Michele Banko, Stephen Soderland and Daniel S. Weld. *Open information extraction from the web*. Comm. of the ACM, pages 68–74, 2008. (Cited on pages 2, 12, 14 and 16.)
- [Ghahramani 2005] Zoubin Ghahramani and Katherine A. Heller. *Bayesian Sets*. In Neural Information Processing Systems, 2005. (Cited on page 13.)
- [Gilleron 2006] Rémi Gilleron, Patrick Marty, Marc Tommasi and Fabien Torre. *Interactive Tuples Extraction from Semi-Structured Data*. In Web Intelligence, pages 997–1004, 2006. (Cited on pages 12, 13, 16 and 45.)
- [Harris 1954] Zellig Harris. *Distributional structure*. Word, vol. 10, pages 146–162, 1954. (Cited on page 12.)
- [Hearst 1992] Marti A. Hearst. *Automatic acquisition of hyponyms from large text corpora*. In Proc. of the Conf. on Computational linguistics, pages 539–545, 1992. (Cited on pages 2 and 12.)
- [Igo 2009] Sean P. Igo and Ellen Riloff. *Corpus-based semantic lexicon induction with Web-based corroboration*. In Proceedings of the Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics, pages 18–26, 2009. (Cited on pages 2, 12 and 13.)

- [Kozareva 2008] Zornitsa Kozareva, Ellen Riloff and Eduard H. Hovy. *Semantic class learning from the Web with hyponym pattern Linkage Graphs*. In Proc. of ACL, pages 1048–1056, 2008. (Cited on pages 2, 10, 12, 14 and 16.)
- [Li 2010] Xiao-Li Li, Lei Zhang, Bing Liu and See-Kiong Ng. *Distributional similarity vs. PU learning for entity set expansion*. In Proceedings of the ACL 2010 Conference Short Papers, page 359—364, 2010. (Cited on page 13.)
- [McDonald 2005] Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin and Pete White. *Simple algorithms for complex relation extraction with applications to biomedical IE*. In Proc. of the An. Meet. on Association for Computational Linguistics, pages 491–498, 2005. (Cited on pages 13, 14, 18 and 78.)
- [Mintz 2009] Mike Mintz, Steven Bills, Rion Snow and Dan Jurafsky. *Distant supervision for relation extraction without labeled data*. In Proc. of the Joint Conf. of the An. Meet. of the ACL and the 4th Int. Joint Conf. on Natural Language Processing of the AFNLP, pages 1003–1011, 2009. (Cited on pages 13, 14 and 16.)
- [Paşca 2007] Marius Paşca. *Weakly-supervised discovery of named entities using web search queries*. In Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pages 683–690, 2007. (Cited on pages 12 and 13.)
- [Pantel 2009] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu and Vishnu Vyas. *Web-scale distributional similarity and entity set expansion*. In Proc. of the Conf. on Empirical Methods in Natural Language Processing, pages 938–947, 2009. (Cited on pages 10, 13, 15 and 16.)
- [Riloff 1997] Ellen Riloff and Jessica Shepherd. *A Corpus-Based Approach for Building Semantic Lexicons*. In Proceedings of the Second Conference on Empirical

- Methods in Natural Language Processing, pages 117–124, 1997. (Cited on page 11.)
- [Talukdar 2006] Partha Pratim Talukdar, Thorsten Brants, Mark Liberman and Fernando Pereira. *A context pattern induction method for named entity extraction*. In Proc. of the Conf. on Computational Natural Language Learning, pages 141–148, 2006. (Cited on pages 2, 10, 14, 16, 55 and 71.)
- [Thelen 2002] Michael Thelen and Ellen Riloff. *A bootstrapping method for learning semantic lexicons using extraction pattern contexts*. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing, pages 214–221, 2002. (Cited on pages 12 and 13.)
- [Turdakov 2010] D. Yu. Turdakov. *Word sense disambiguation methods*. Program. Comput. Softw., pages 309–326, 2010. (Cited on page 16.)
- [Wang 2007] Richard C. Wang and William W. Cohen. *Language-independent set expansion of named entities using the Web*. In Proc. of the IEEE Int. Conf. on Data Mining, pages 342–350, 2007. (Cited on pages v, vi, viii, 6, 9, 12, 15, 16, 17, 19, 25, 26, 30, 31, 32 and 40.)
- [Wang 2008] R. C. Wang and W. W. Cohen. *Iterative set expansion of named entities using the Web*. In Proc. of the IEEE Int. Conf. on Data Mining, pages 1091–1096, 2008. (Cited on pages 2, 10, 13, 16, 51, 55 and 71.)
- [Wang 2009] R. C. Wang and W. W. Cohen. *Character-level analysis of semi-structured documents for set expansion*. In Proc. of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1503–1512, 2009. (Cited on pages vi, viii, 9, 13, 16, 28, 29, 32, 33 and 61.)
- [Widdows 2002] Dominic Widdows and Beate Dorow. *A graph model for unsupervised lexical acquisition*. In Proceedings of the 19th international conference on Computational linguistics, pages 1–7, 2002. (Cited on pages 11 and 13.)

- [Zhang 2011] Lei Zhang and Bing Liu. *Entity set expansion in opinion documents*. In Proceedings of the 22nd ACM conference on Hypertext and hypermedia, pages 281–290, 2011. (Cited on pages 2 and 13.)

Datasets Description and Results

Illustration

In this section, we summarize each dataset from the goal and task to the experimental results, such as the top 20 candidate t-uples, top 10 domains, top 20 Web pages. Note that all the experimental results illustrated in this section are returned by our STEP with parameter setting as follows.

Parameter	Value
I	1
N_c	20
N_p	100
N_s	2
<i>siblingFlag</i>	false

Table A.1: Parameter setting of STEP.

A.1 D1

Task. Given a set of examples, e.g., $\{ \langle \textit{Amateur Radio, India} \rangle, \langle \textit{CQ-PA, Netherlands} \rangle \}$, the goal is to extract a list of instances of a binary relation $\langle \textit{Amateur Radio Magazine, Country of Origin} \rangle$, i.e., pairs of amateur radio magazines and their countries of origin.

Top 20 candidate t-uples. (1) $\langle \textit{cq-pa, netherlands} \rangle$ (2) $\langle \textit{amateur radio, india} \rangle$ (3) $\langle \textit{p a d i o a m a t o p, ukraine} \rangle$ (4) $\langle \textit{electron, netherlands} \rangle$ (5) $\langle \textit{radiocomunicaii i radioamatorism, romania} \rangle$ (6) $\langle \textit{radiorivista, italy} \rangle$ (7) $\langle \textit{wiat}$

radio,poland> (8) <*electron,netherlands.svg/22px-flag of the netherlands*> (9)
 <*hamformosa,taiwan*> (10) <*hamformosa,republic of china.svg/22px-flag of the
 republic of china*> (11) <*radio news,the united states*> (12) <*hellenic radio
 magazine,greece*> (13) <*radioamateur magazine,france*> (14) <*radio ref,france*>
 (15) <*electron,the netherlands*> (16) <*modern electricians,the united states*> (17)
 <*cq communications today,the people's republic of china*> (18) <*radio, fernsehen,
 elektronik,germany*> (19) <*cq dl,germany*> (20) <*electronics illustrated,the united
 states*>.

Top ten domains. (1) www.massmediadistribution.com (2) www.mshtawy.com
 (3) www.territorioscuola.com (4) pediaview.com (5) www.ask.com (6) uk.ask.com
 (7) www.rescue.kate-jenter.com (8) www.house.giftedamersexdating.com
 (9) www.r-domain.net (10) www.eqsl.cc.

Top ten Web pages. (1) www.mshtawy.com/en-wiki.php?title=List_of_amateur_radio_magazines (2) wikiand.com/wiki/List_of_amateur_radio_magazines (3) pediaview.com/openpedia/List_of_amateur_radio_magazines (4) www.territorioscuola.com/wikipedia/en.wikipedia.php?title=List_of_amateur_radio_magazines (5) www.secret-bases.co.uk/wiki.php?url=wiki/List_of_amateur_radio_magazines (6) www.ask.com/wiki/p-List_of_amateur_radio_magazines (7) www.rescue.kate-jenter.com/p-List_of_amateur_radio_magazines (8) www.house.giftedamersexdating.com/List_of_amateur_radio_magazines (9) uk.ask.com/wiki/List_of_amateur_radio_magazines (10) abitabout.com/List+of+amateur+radio+magazines.

A.2 D2

Task. Given a set of examples, e.g., {<*Liberia,18.3*>, <*Chad,15.4*>}, the goal is to extract a list of instances of a binary relation <*Country, Death Rate*>, i.e., pairs of countries and their death rates.

Top 20 candidate t-uples. (1) *<liberia,18.3>* (2) *<chad,15.4>* (3) *<germany,10.90>* (4) *<israel,5.5>* (5) *<india,6.23>* (6) *<norway,9.1>* (7) *<sao,tome and principe 7.5>* (8) *<burkina,faso 14.4>* (9) *<papua,new guinea 9.6>* (10) *<virgin,islands 6.6>* (11) *<benin,11.2>* (12) *<cyprus,7.5>* (13) *<isle of man,10.76>* (14) *<the isle of man,10.76>* (15) *<andorra,the 5.89>* (16) *<romania,11.88>* (17) *<trinidad and tobago,8.11>* (18) *<yemen,7.61>* (19) *<djibouti,19.10>* (20) *<tunisia,5.6>*.

Top ten domains. (1) www.unctad.org (2) www.telecomservices.net (3) www.fawe.org (4) www.holmatro.com (5) earthtrends.wri.org (6) prepaid-calling-card.phonebestcard.com (7) www.88card.com (8) www.vipvoip.nl (9) www.un.org (10) www.statcompiler.org.

Top ten Web pages. (1) www.cheapbeninphonecard.com/country-list.html (2) www.shashiservices.in/submersible-pumps.htm (3) www.layatel.com/u/from-india.html (4) www.statcompiler.org/tableBuilderController.cfm?tables=87&survey_ids=147,248&table_orientation=R&fromSurveyList=quickstats&CFID=13940176&CFTOKEN=90499327 (5) www.zeropin.com/php/web/rate.php (6) www.mundomanz.com/meteo_p/main?l=1 (7) www.fawe.org/region/east/uganda/index.php (8) www.teleaccount.com/PriceList.aspx (9) www.mvpei.hr/MVP.asp?pcpid=1621 (10) www.iran-phone-card.com/country-list.html.

A.3 D3

Task. Given a set of examples, e.g., $\{ \langle ARB, Administrative Review Board \rangle, \langle VOA, Voice of America \rangle \}$, the goal is to extract a list of instances of a binary relation $\langle US Agency Abbreviation, Full Name \rangle$, i.e., pairs of the US agency abbreviations and their full names.

Top 20 candidate t-uples. (1) *<voa, voice of america>* (2) *<arb, administrative review board>* (3) *<aid, agency for international development>* (4) *<cea, coun-*

cil of economic advisers> (5) <*cdfi, community development financial institutions*> (6) <*cen, bureau of the census*> (7) <*cia, central intelligence agency*> (8) <*ceppo, chemical emergency preparedness and prevention office*> (9) <*cic, consumer information center*> (10) <*achp, advisory council on historic preservation*> (11) <*ahrq, agency for healthcare research and quality*> (12) <*ceq, council on environmental quality*> (13) <*niams, national institute of arthritis and musculoskeletal and skin diseases*> (14) <*afspc, air force space command*> (15) <*cfoc, chief financial officers council*> (16) <*ahcpr, agency for health care policy and research*> (17) <*cetec, topographic engineering center*> (18) <*cfsan, national center for food safety and applied nutrition*> (19) <*acsl, alternate crops and systems lab*> (20) <*who, world health organization*>.

Top ten domains. (1) www.egloballibrary.com (2) www.solveariddle.com (3) www.absoluteastronomy.com (4) www.njcarinsurance.org (5) www.turbobuicks.com (6) post_119_gulfport_ms.tripod.com (7) www.acronymlist.org (8) www.acronymdict.com (9) liberalforum.org (10) bbs.1000fr.net.

Top ten Web pages. (1) wn.com/Guantanamo_military_commission (2) www.fedjobs.com/chat/agency_acronymns.html (3) www.solveariddle.com/coolacronyms/acronym.php?cat=US%20Govt.%20Acronyms (4) www.egloballibrary.com/egl/html/LinkBot/DynamicLinkChecker.html (5) puls.se/Many-Pakistanis-still-waiting-for-flood-aid-Afghanistan-Relief-Organization-lhjSwPw4owJS (6) www.assignedriskauto.org/us-gov-abbreviations-acronyms.htm (7) www.acronymlist.org/acronym/VOA-42083.html (8) data.govloop.com/api/views/f2gs-6w6p/rows.pdf?app_token=U29jcmFOYS0td2VraWNrYXNz0 (9) www.njcarinsurance.org/US-Gov-Acronyms-websites.htm (10) www.historycommons.org/topic.jsp?startpos=900&topic=topic_imperialism_and_domination.

A.4 D4

Task. Given a set of examples, e.g., $\{ \langle \textit{Austria, States of Austria} \rangle, \langle \textit{Iraq, Governorates of Iraq} \rangle \}$, the goal is to extract a list of instances of a binary relation $\langle \textit{Federation, Federating Units} \rangle$, i.e., pairs of federation and their federating units.

Top 20 candidate t-uples. (1) $\langle \textit{austria, states of austria} \rangle$ (2) $\langle \textit{iraq, governorates of iraq} \rangle$ (3) $\langle \textit{brazil, states of brazil} \rangle$ (4) $\langle \textit{mexico, states of mexico} \rangle$ (5) $\langle \textit{sudan, states of sudan} \rangle$ (6) $\langle \textit{venezuela, states of venezuela} \rangle$ (7) $\langle \textit{nigeria, states of nigeria} \rangle$ (8) $\langle \textit{germany, states of germany} \rangle$ (9) $\langle \textit{malaysia, states of malaysia} \rangle$ (10) $\langle \textit{lebanon, governorates of lebanon} \rangle$ (11) $\langle \textit{kuwait, governorates of kuwait} \rangle$ (12) $\langle \textit{egypt, governorates of egypt} \rangle$ (13) $\langle \textit{palau, states of palau} \rangle$ (14) $\langle \textit{micronesia, states of the federated states of micronesia} \rangle$ (15) $\langle \textit{the united arab emirates, emirates of the united arab emirates} \rangle$ (16) $\langle \textit{jordan, governorates of jordan} \rangle$ (17) $\langle \textit{syria, governorates of syria} \rangle$ (18) $\langle \textit{yemen, governorates of yemen} \rangle$ (19) $\langle \textit{tunisia, governorates of tunisia} \rangle$ (20) $\langle \textit{bahrain, governorates of bahrain} \rangle$.

Top ten domains. (1) www.absoluteastronomy.com (2) districtplace.com (3) tmp.kiwix.org:4201 (4) www.weidia.com (5) districtenrollment.com (6) www.scribd.com (7) wapedia.mobi (8) www.nationmaster.com (9) www.xklsv.org (10) commons.wikimedia.org.

Top ten Web pages. (1) commons.wikimedia.org/wiki/Atlas_of_first-level_administrative_divisions (2) www.netipedia.com/index.php/Wikipedia:Navigational_templates (3) wn.com/federated_state?orderby=relevance (4) wapedia.mobi/en/Category:First-level_administrative_country_subdivisions (5) tmp.kiwix.org:4201/A/Federation.html (6) www.absoluteastronomy.com/topics/District (7) www.nationmaster.com/encyclopedia/List-of-FIPS-region-codes (8) districtplace.com/ (9) districtenrollment.com/ (10) www.weidia.com/en-wiki/Federation.

A.5 D5

Task. Given a set of examples, e.g., $\{\langle \textit{Algeria}, \textit{ALG} \rangle, \langle \textit{Andorra}, \textit{AND} \rangle\}$, the goal is to extract a list of instances of a binary relation $\langle \textit{Country}, \textit{FIFA Code} \rangle$, i.e., pairs of countries and their FIFA codes.

Top 20 candidate t-uples. (1) $\langle \textit{algeria}, \textit{alg} \rangle$ (2) $\langle \textit{andorra}, \textit{and} \rangle$ (3) $\langle \textit{albania}, \textit{alb} \rangle$ (4) $\langle \textit{armenia}, \textit{arm} \rangle$ (5) $\langle \textit{austria}, \textit{aut} \rangle$ (6) $\langle \textit{argentina}, \textit{arg} \rangle$ (7) $\langle \textit{afghanistan}, \textit{afg} \rangle$ (8) $\langle \textit{angola}, \textit{ang} \rangle$ (9) $\langle \textit{australia}, \textit{aus} \rangle$ (10) $\langle \textit{aruba}, \textit{aru} \rangle$ (11) $\langle \textit{american samoa}, \textit{asa} \rangle$ (12) $\langle \textit{azerbaijan}, \textit{aze} \rangle$ (13) $\langle \textit{anguilla}, \textit{aia} \rangle$ (14) $\langle \textit{antigua}, \textit{ant} \rangle$ (15) $\langle \textit{netherlands}, \textit{ned} \rangle$ (16) $\langle \textit{india}, \textit{ind} \rangle$ (17) $\langle \textit{antigua}, \textit{and barbuda}, \textit{ant} \rangle$ (18) $\langle \textit{canada}, \textit{can} \rangle$ (19) $\langle \textit{belgium}, \textit{bel} \rangle$ (20) $\langle \textit{georgia}, \textit{geo} \rangle$.

Top ten domains. (1) uk.ask.com (2) www.weather2flights.com (3) www.pwc.com (4) www.quadrodemedalhas.com (5) www.arrs.net (6) www.iomclass.org (7) www.daviscup.com (8) www.yasni.com (9) www.soccerygaming.tv (10) www.docstoc.com.

Top ten Web pages. (1) www.oocities.org/tds_founder/iwufmembers.htm (2) www.bingohideout.co.uk/all-you-need-to-know-about-the-olympic-games.html (3) www.tm-forum.com/viewtopic.php?f=124&t=16627&start=195 (4) www.eccma.org.in/NewMemberApplication.php (5) www.gamescampaign.com/register.php (6) www.clicksrnk.com/register.php (7) www.hostadz.com/register.php (8) www.amaneo-ads.com/register.php (9) www.adquick.co.uk/register.php/ (10) www.docstoc.com.

A.6 D6

Task. Given a set of examples, e.g., $\{\langle \textit{新奥尔良黄蜂}, \textit{New Orleans Hornets} \rangle, \langle \textit{芝加哥公牛}, \textit{Chicago Bulls} \rangle\}$, the goal is to extract a list of instances of a binary relation $\langle \textit{NBA Team in Chinese}, \textit{NBA Team in English} \rangle$, i.e., pairs of NBA team names in Chinese and that in English.

Top 20 candidate t-uples. (1) <新奥尔良黄蜂, *new orleans hornets*> (2) <芝加哥公牛, *chicago bulls*> (3) <达拉斯小牛, *dallas mavericks*> (4) <圣安东尼奥马刺, *san antonio spurs*> (5) <丹佛掘金, *denver nuggets*> (6) <波士顿凯尔特人, *boston celtics*> (7) <波特兰开拓者, *portland trail blazers*> (8) <密尔沃基雄鹿, *milwaukee bucks*> (9) <迈阿密热火, *miami heat*> (10) <克利夫兰骑士, *cleveland cavaliers*> (11) <菲尼克斯太阳, *phoenix suns*> (12) <休斯敦火箭, *houston rockets*> (13) <亚特兰大老鹰, *atlanta hawks*> (14) <多伦多猛龙, *toronto raptors*> (15) <萨克拉门托国王, *sacramento kings*> (16) <克里夫兰骑士, *cleveland cavaliers*> (17) <奥兰多魔术, *orlando magic*> (18) <新泽西篮网, *new jersey nets*> (19) <纽约尼克斯, *new york knicks*> (20) <犹他爵士, *utah jazz*>.

Top ten domains. (1) <https://picasaweb.google.com> (2) zhidao.baidu.com (3) www.pickhoody.com (4) www.yaomingmania.com (5) picasaweb.google.com (6) www.huanantiger.com (7) uk.androlib.com (8) www.shopadidastw.com (9) tvboxnow.com (10) dbpedia.org.

Top ten Web pages. (1) www.kobechina.com.cn/archiver/tid-37287.html (2) fr.appbrain.com/app/nba2011-all-star-wonderful-pic/com.nba2011 (3) www.powderbomb.com/coppermine/displayimage.php?album=4&pos=12 (4) picasaweb.google.com/cutebizok22 (5) zhidao.baidu.com/question/113424527.html (6) www.androlib.com/android.application.com-nba2011-pAtAx.aspx (7) www.appbrain.com/app/nba2011-all-star-wonderful-pic/com.nba2011 (8) market.android.com/details?id=com.nba2011 (9) bbs.66xue.com/viewthread.php?action=printable&tid=22660 (10) www.huanantiger.com/archiver/?tid-28203.html.

A.7 D7

Task. Given a set of examples, e.g., {<*Aeolian Vision*,9483554>, <*Aegean Star*,7502942>}, the goal is to extract a list of instances of a binary relation <*Ship Name*, *IMO*>, i.e., pairs of ships and their IMO numbers.

Top 20 candidate t-uples. (1) *<aeolian vision, 9483554>* (2) *<aegean star, 7502942>* (3) *<aegean pearl, 7722621>* (4) *<aeolian heritage, 9483542>* (5) *<aeolos, 6524060>* (6) *<aeolis, 7614525>* (7) *<aegean wind, 8130746>* (8) *<aetos, 9225524>* (9) *<agonistis, 9495715>* (10) *<anangel dawn, 9455533>* (11) *<agios nikolas, 9291779>* (12) *<agia marina, 7710032>* (13) *<achilleus, 7377464>* (14) *<alpha prudence, 9423762>* (15) *<alexandria, 8004181>* (16) *<alpha friendship, 9123374>* (17) *<amethyst, 9323132>* (18) *<anangel explorer, 9295012>* (19) *<amalia, 9180906>* (20) *<anangel guardian, 9434369>*.

Top ten domains. (1) www.hafenradar.de (2) www.vesseltracker.com (3) (4) (5) (6) (7) (8) (9) (10) .

(Note there exist only two domains in this case.)

Top ten Web pages. (1) www.hafenradar.de/en/vessels?page=11 (2) www.vesseltracker.com/en/VesselArchive/All/Cargo-ships.html?search=0+&tablePage=8 (3) www.vesseltracker.com/en/VesselArchive/All/Cargo-ships.html?search=H+&tablePage=8 (4) www.vesseltracker.com/en/VesselArchive/All/Cargo-ships.html?search=J+&tablePage=8 (5) www.vesseltracker.com/en/VesselArchive/All/Cargo-ships.html?search=A+&tablePage=8 (6) www.vesseltracker.com/en/VesselArchive/All/Cargo-ships.html?search=X+&tablePage=8 (7) www.vesseltracker.com/en/VesselArchive/All/Cargo-ships.html?search=W+&tablePage=8 (8) www.vesseltracker.com/en/VesselArchive/Greece/Cargo-ships.html?search=A (9) www.vesseltracker.com/en/VesselArchive/All/Cargo-ships.html?search=P+&tablePage=8 (10) www.vesseltracker.com/en/VesselArchive.html?country=Greece&search=C+&shipType=cargo_ships&sortColumn=name.

A.8 D8

Task. Given a set of examples, e.g., $\{ \langle CG1101, Programming Methodology \rangle, \langle CP3201, Industry Seminar \rangle \}$, the goal is to extract a list of instances of a bi-

nary relation $\langle NUS\ SoC\ Module\ code, NUS\ SoC\ Module\ Name \rangle$, i.e., pairs of NUS Soc module codes and corresponding module names.

Top 20 candidate t-uples. (1) $\langle cg1101, programming\ methodology \rangle$ (2) $\langle cp3201, industry\ seminar \rangle$ (3) $\langle cg1103, data\ structures\ and\ algorithms\ i \rangle$ (4) $\langle cs3225, combinatorial\ methods\ in\ bioinformatics \rangle$ (5) $\langle cs5342, multimedia\ computing\ and\ applications \rangle$ (6) $\langle cs1281, c\ to\ java \rangle$ (7) $\langle cs4271, critical\ systems\ and\ their\ verification \rangle$ (8) $\langle cs5226, database\ tuning \rangle$ (9) $\langle cs5239, computer\ system\ performance\ analysis \rangle$ (10) $\langle cs5218, principles\ of\ program\ analysis \rangle$ (11) $\langle cs5230, computational\ complexity \rangle$ (12) $\langle cs5240, theoretical\ foundations\ in\ multimedia \rangle$ (13) $\langle cs6208, advanced\ topics\ in\ artificial\ intelligence \rangle$ (14) $\langle cs2106, introduction\ to\ operating\ systems \rangle$ (15) $\langle cs4236, cryptography\ theory\ and\ practice \rangle$ (16) $\langle cs5322, database\ security \rangle$ (17) $\langle cs4211, formal\ methods\ for\ software\ engineering \rangle$ (18) $\langle cs5241, speech\ processing \rangle$ (19) $\langle cs3248, design\ of\ interactive\ systems \rangle$ (20) $\langle it1006, matlab\ programming\ for\ mathematics \rangle$.

Top ten domains. (1) www.docstoc.com (2) <https://sit.aces01.nus.edu.sg> (3) www.comp.nus.edu.sg (4) <https://aces01.nus.edu.sg> (5) (6) (7) (8) (9) (10) .

(Note there exist only four domains in this case.)

Top ten Web pages. (1) www.docstoc.com/docs/79584469/Course-Description---NUS---School-of-Computing (2) <https://sit.aces01.nus.edu.sg/cors/jsp/report/ModuleInfoListing.jsp> (3) www.comp.nus.edu.sg/undergraduates/useful_course_schedule.html (4) <https://aces01.nus.edu.sg/cors/jsp/report/ModuleInfoListing.jsp> (5) www.comp.nus.edu.sg/~online/course_list/coursesNL.html (6) www.comp.nus.edu.sg/~online/course_list/courses.html (7) <https://aces01.nus.edu.sg/cors/jsp/report/ModuleInfoListing.jsp?...> (8) www.docstoc.com/docs/82453894/MODULES-OFFERED-BY-SCHOOL-OF-COMPUTING-C (9) (10) .

(Note only eight pages that contain the seeds are returned by the search engine.)

A.9 D9

Task. Given a set of examples, e.g., {<桃园县,朱立伦>, <金门县,李炆烽>}, the goal is to extract a list of instances of a binary relation <Taiwan City in Chinese, Mayor in Chinese>, i.e., pairs of Taiwan cities in Chinese and their mayors in Chinese.

Top 20 candidate t-uples. (1) <金门县, 李炆烽> (2) <桃园县, 朱立伦> (3) <苗栗县, 刘政鸿> (4) <新竹县, 郑永金> (5) <台南县, 苏焕智> (6) <高雄县, 杨秋兴> (7) <连江县, 陈雪生> (8) <南投县, 李朝卿> (9) <台中市, 胡志强> (10) <嘉义县, 陈明文> (11) <台中县, 黄仲生> (12) <花莲县, 谢深山> (13) <台南市, 许添财> (14) <新竹市, 林政则> (15) <基隆市, 许财利> (16) <台东县, 吴俊立> (17) <屏东县, 曹启鸿> (18) <彰化县, 卓伯源> (19) <嘉义市, 黄敏惠> (20) <云林县, 苏治芬>.

Top ten domains. (1) www.cnrr.cn (2) tw.people.com.cn (3) www.chinaelections.org (4) gb.udndata.com (5) www.nhaidu.com (6) house.focus.cn (7) bbs.gd.gov.cn (8) news.upc.edu.cn (9) info.cndsi.com (10) bbs.tiexue.net.

Top ten Web pages. (1) news.eastday.com/epublish/gb/paper139/46/class013900018/hwz549889.htm (2) wapedia.mobi/zh/2001%E5%B9%B4%E4%B8%AD%E8%8F%AF%E6%B0%91%E5%9C%8B%E7%B8%A3%E5%B8%82%E9%95%B7%E9%81%B8%E8%88%89 (3) www.xici.net/b567996/d32837231.htm (4) www.tianshui.com.cn/news/guonei/2005120322413438772.htm (5) news.xinhuanet.com/tai_gang_ao/2005-12/04/content_3873717.htm (6) www.wjppjzx.net/Article/ShowArticle.asp?ArticleID=53 (7) www.cetin.net.cn/cetin2/servlet/cetin/action/HtmlDocumentAction;jsessionId=669980967DC48C74C70086803033B085?baseid=1&docno=254756 (8) www.wzqjks.com/Article_Print.asp?ArticleID=2094 (9) bbs.zxrs.net/dispbbs_69_60814_1_5.html (10) www.cnrr.cn/cnrrjs/shownews.asp?newsid=990&name=%BA%A3%CF%BF%C1%BD%B0%B6.

A.10 D10

Task. Given a set of examples, e.g., {<John Baldacci, Maine>, <Arnold Schwarzenegger, California>}, the goal is to extract a list of instances of a binary relation <US State Governor, US State>, i.e., pairs of US state governors and corresponding US states.

Top 20 candidate t-uples. (1) <arnold schwarzenegger, california> (2) <john baldacci, maine> (3) <bill richardson, new mexico> (4) <jennifer granholm, michigan> (5) <donald l. carciere, rhode island> (6) <bob riley, alabama> (7) <sonny perdue, georgia> (8) <brad henry, oklahoma> (9) <martin o'malley, maryland> (10) <james douglas, vermont> (11) <bill ritter, colorado> (12) <deval patrick, massachusetts> (13) <donald carciere, rhode island> (14) <mark sanford, south carolina> (15) <eliot spitzer, new york> (16) <christine gregoire, washington state> (17) <phil bredeesen, tennessee> (18) <ed rendell, pennsylvania> (19) <janet napolitano, arizona> (20) <jodi rell, connecticut>.

Top ten domains. (1) www.toledoblade.com (2) statehouserock.com (3) theenergycollective.com (4) www.renewablechoice.com (5) wapedia.mobi (6) www.freerepublic.com (7) www.eesf.org (8) onenationundergod.org (9) www.massnews.com (10) oc-divorce.typepad.com.

Top ten Web pages. (1) www.theday.com/article/20110603/NWS12/306039919 (2) thealternativepress.com/articles/governor-corzine-ranks-in-the-top-ten-green-governors (3) www.thepoliticalweekly.blogspot.com/ (4) www.grist.org/article/schwarzeneggers-response (5) www.realclearpolitics.com/Commentary/com-2_3_06_LS.html (6) www.newenglandfutures.org/issues/energy/article/ (7) www.onlinesentinel.com/news/stars-light-up-schools-wall-of-fame_2011-06-03.html?pageType=mobile&id=1 (8) www.freerepublic.com/focus/f-gop/1009268/posts (9) onenationundergod.org/wl_gaymarriage.html (10) www.corpwatch.org/article.php?id=14546.

A.11 D11

Task. Given a set of examples, e.g., $\{\langle \textit{NotePub}, \textit{General purpose} \rangle, \langle \textit{TermWiki}, \textit{Terminology management platform} \rangle\}$, the goal is to extract a list of instances of a binary relation $\langle \textit{Wiki}, \textit{Focus} \rangle$, i.e., pairs of wikis and their focus.

Top 20 candidate t-uples. (1) $\langle \textit{termwiki}, \textit{terminology management platform} \rangle$ (2) $\langle \textit{notepub}, \textit{general purpose} \rangle$ (3) $\langle \textit{wikihow}, \textit{general instruction} \rangle$ (4) $\langle \textit{geonames}, \textit{places} \rangle$ (5) $\langle \textit{international music score library project}, \textit{music} \rangle$ (6) $\langle \textit{snpedia}, \textit{science—biology} \rangle$ (7) $\langle \textit{susning.nu}, \textit{encyclopedic /swedish} \rangle$ (8) $\langle \textit{baidu baike}, \textit{encyclopedic /chinese} \rangle$ (9) $\langle \textit{quora}, \textit{general knowledge} \rangle$ (10) $\langle \textit{lyricwiki}, \textit{music—lyrics} \rangle$ (11) $\langle \textit{congresspedia}, \textit{government—united states congress} \rangle$ (12) $\langle \textit{uncyclopedia}, \textit{satire—parody} \rangle$ (13) $\langle \textit{wikimedia commons}, \textit{misc—electronic media} \rangle$ (14) $\langle \textit{travellerspoint}, \textit{places—travel} \rangle$ (15) $\langle \textit{wikiquote}, \textit{reference—quotations} \rangle$ (16) $\langle \textit{knol}, \textit{general purpose} \rangle$ (17) $\langle \textit{foodista.com}, \textit{reference—food and cooking} \rangle$ (18) $\langle \textit{wikidia}, \textit{encyclopedic /spanish and french} \rangle$ (19) $\langle \textit{wikicandidate}, \textit{fictional presidential campaign} \rangle$ (20) $\langle \textit{mywikibiz}, \textit{misc—business directory} \rangle$.

Top ten domains. (1) dikkedeur.dyndns.org (2) nonpedia.org (3) e-gold.dreab.com (4) olgakuvaiskova.wikispaces.com (5) www.zeszytowy.travel.pl (6) www.ohio.e90.biz (7) wiki.verkata.com (8) mashpedia.com (9) www.kentucky.e90.biz (10) www.top40-charts.info.

Top ten Web pages. (1) wn.com/list_of_wikis?orderby=relevance&upload_time=all_time (2) www.dotleb.net/directory/index.php?title=List_of_wikis (3) www.freebooknotes.com/wiki/List_of_wikis (4) danpritchard.com/wiki/List_of_wikis (5) www.territorioscuola.com/wikipedia/en.wikipedia.php?title=List_of_wikis (6) wiki.verkata.com/en/wiki/List_of_wikis (7) www.pricing.giftedamersexdating.com/p-List_of_wikis (8) nonpedia.org/wiki/List_of_wikis (9) www.healthcare.reachinformation.com/List%20of%20wikis.aspx (10) www.wholesale.giftedamersexdating.com/p-List_of_wikis.

A.12 D12

Task. Given a set of examples, e.g., $\{ \langle \textit{Germany}, 931,000,000,000, 2009 \rangle, \langle \textit{Russia}, 302,000,000,000, 2008 \rangle \}$, the goal is to extract a list of instances of a triple relation $\langle \textit{Country}, \textit{Import}, \textit{Year} \rangle$, i.e., a list of countries, and the imports in some year.

Top 20 candidate t-uples. (1) $\langle \textit{russia}, 302,000,000,000, 2008 \rangle$
 (2) $\langle \textit{germany}, 931,000,000,000, 2009 \rangle$ (3) $\langle \textit{iraq}, 43,500,000,000, 2008 \rangle$
 (4) $\langle \textit{namibia}, 3,560,000,000, 2008 \rangle$ (5) $\langle \textit{cyprus}, 8,689,000,000, 2008 \rangle$
 (6) $\langle \textit{gabon}, 2,830,000,000, 2008 \rangle$ (7) $\langle \textit{venezuela}, 53,440,000,000, 2008 \rangle$
 (8) $\langle \textit{malawi}, 1,023,000,000, 2008 \rangle$ (9) $\langle \textit{bangladesh}, 20,170,000,000, 2008 \rangle$
 (10) $\langle \textit{syria}, 14,320,000,000, 2008 \rangle$ (11) $\langle \textit{sudan}, 7,757,000,000, 2008 \rangle$ (12)
 $\langle \textit{chad}, 1,470,000,000, 2008 \rangle$ (13) $\langle \textit{jamaica}, 7,191,000,000, 2008 \rangle$ (14)
 $\langle \textit{serbia}, 22,875,000,000, 2008 \rangle$ (15) $\langle \textit{uruguay}, 7,000,000,000, 2008 \rangle$ (16)
 $\langle \textit{madagascar}, 2,541,000,000, 2008 \rangle$ (17) $\langle \textit{cameroon}, 4,362,000,000, 2008 \rangle$ (18)
 $\langle \textit{slovenia}, 38,120,000,000, 2008 \rangle$ (19) $\langle \textit{palau}, 107,300,000, 2004 \rangle$ (20) $\langle \textit{benin}, 1,355,000,000, 2008 \rangle$.

Top ten domains. (1) [wapedia.mobi](#) (2) [wn.com](#) (3) [www.enotes.com](#) (4) [enc.tfode.com](#) (5) [www.razorrobotics.com](#) (6) (7) (8) (9) (10) .

(Note there exist only five domains in this case.)

Top ten Web pages. (1) [wapedia.mobi/en/List_of_countries_by_imports?t=1](#). (2) [wn.com/list_of_countries_by_imports?orderby=relevance&upload_time=all_time](#) (3) [www.enotes.com/topic/List_of_countries_by_imports](#) (4) [enc.tfode.com/List_of_countries_by_imports](#) (5) [www.razorrobotics.com/knowledge/?title=List_of_countries_by_imports](#) (6) [wn.com/list_of_countries_by_imports?orderby=relevance](#) (7) [wn.com/list_of_countries_by_imports?upload_time=all_time&orderby=rating](#) (8) [wn.com/list_of_countries_by_imports](#) (9) [wn.com/list_of_countries_by_imports?orderby=rating](#) (10).

(Note only nine pages that contain the seeds are returned by the search engine.)

A.13 D13

Task. Given a set of examples, e.g., {<Lucky Number,Lam Po Ko, DS Movie Production>, <Sharp Pencil,Gallen Mei, Under Pressure Pictures>}, the goal is to extract a list of instances of a triple relation <Singapore Film, Director, Producer>, i.e., a list of Singapore films, and the directors and producers of the films.

Top 20 candidate t-uples. (1) <lucky number, lam po ko, ds movie production> (2) <sharp pencil, gallen mei, under pressure pictures> (3) <a road less travelled, lim suat yen, oak 3 films> (4) <the teenage textbook movie, philip lim, monster films> (5) <city sharks, esan sivalingam, hoodsinc productions> (6) <stamford hall, manoharan ramakrishnan, temasek hall , nus> (7) <god or dog, hugo ng, l s entertainment> (8) <2003, city sharks, esan sivalingam> (9) <smell of rain, gloria chee, smell of rain production> (10) <clouds in my coffee, gallen mei, reversal films> (11) <one last dance, max makowski, ming productions> (12) <anna & anna, aubrey lam, ng sian ngoh> (13) <2000, stories about love, cheek> (14) <avatar, kuo jian hong, cinemancer> (15) <one leg kicking, wei koh / eric khoo, zhao wei films / raintree pictures / sfc> (16) <zombie dogs, toh hai leong, zhao wei films> (17) <2001, sharp pencil, gallen mei> (18) <the kallang wave, yan-feng lee / hanafi ramdan, zayed bin abdul aziz talib> (19) <bugis street, yon fan, jaytex productions> (20) <tiger's whip, victor khoo, river films>.

Top ten domains. (1) www.nethelper.com (2) www.servinghistory.com (3) www.nationmaster.com (4) mashpedia.com (5) wapedia.mobi (6) pediaview.com (7) wpedia.goo.ne.jp (8) maps.thefullwiki.org (9) medlibrary.org (10) www.peach.dreab.com.

Top ten Web pages. (1) wn.com/List_of_Singaporean_films (2) www.globalwarmingart.com/wiki/Wikipedia:List_of_Singaporean_films (3) nl.wikitu.com/wiki/List_of_Singaporean_films (4) www.digparty.com/wiki/List_of_Singaporean_films (5) winx.name/?w=List_of_Singaporean_films (6) wikian.d.com/wiki/List_of_Singaporean_films (7) www.grooveasia.com/directory/p

age/List_of_Singaporean_films (8) [zml.name/?z=List_of_Singaporean_films](#)
 (9) [www.abitabout.com/List+of+Singaporean+films](#) (10) [www.reference.com/browse/derrol](#).

A.14 D14

Task. Given a set of examples, e.g., $\{ \langle \text{Czech Republic}, 10,535,811, \text{March 31}, 2011, 0.15\% \rangle, \langle \text{Ethiopia}, 82,101,998, 2011, 1.18\% \rangle \}$, the goal is to extract a list of instances of a 4-ary relation $\langle \text{Country}, \text{Population}, \text{Date}, \text{percentage of World population} \rangle$, i.e., a list of countries, the populations of the countries on some day, and the percentage of the world population.

Top 20 candidate t-uples. (1) $\langle \text{ethiopia}, 82,101,998, 2011, 1.18\% \rangle$ (2) $\langle \text{czech republic}, 10,535,811, \text{march 31}, 2011, 0.15\% \rangle$ (3) $\langle \text{trinidad and tobago}, 1,317,714, \text{july 1}, 2010, 0.019\% \rangle$ (4) $\langle \text{panama}, 3,405,813, \text{may 16}, 2010, 0.049\% \rangle$ (5) $\langle \text{argentina}, 40,091,359, \text{october 27}, 2010, 0.58\% \rangle$ (6) $\langle \text{united states}, 311,887,000, \text{august 3}, 2011, 4.5\% \rangle$ (7) $\langle \text{netherlands}, 16,686,600, \text{august 3}, 2011, 0.241\% \rangle$ (8) $\langle \text{pakistan}, 176,815,000, \text{august 3}, 2011, 2.55\% \rangle$ (9) $\langle \text{india}, 1,210,193,422, \text{march 1}, 2011, 17.45\% \rangle$ (10) $\langle \text{mauritius}, 1,280,925, \text{july 1}, 2010, 0.018\% \rangle$ (11) $\langle \text{estonia}, 1,340,122, \text{july 1}, 2011, 0.019\% \rangle$ (12) $\langle \text{burkina faso}, 15,730,977, \text{july 1}, 2010, 0.23\% \rangle$ (13) $\langle \text{seychelles}, 86,525, \text{july 1}, 2010, 0.001\% \rangle$ (14) $\langle \text{cameroon}, 19,406,100, \text{july 1}, 2010, 0.28\% \rangle$ (15) $\langle \text{liechtenstein}, 36,157, \text{december 31}, 2010, 0.0005\% \rangle$ (16) $\langle \text{cuba}, 11,241,161, \text{december 31}, 2010, 0.16\% \rangle$ (17) $\langle \text{cape verde}, 491,575, \text{june 16}, 2010, 0.007\% \rangle$ (18) $\langle \text{algeria}, 36,300,000, \text{july 1}, 2011, 0.52\% \rangle$ (19) $\langle \text{uruguay}, 3,356,584, \text{june 30}, 2010, 0.048\% \rangle$ (20) $\langle \text{republic of ireland}, 4,581,269, \text{april 10}, 2011, 0.066\% \rangle$.

Top ten domains. (1) [plumbot.com](#) (2) [www.youramazingsmile.com](#) (3) [wapedia.mobi](#) (4) [www.peoplesrepublicofcork.com](#) (5) [www.wiki.networkbase.info](#) (6) [www.maxcaratulas.net](#) (7) [www.poker.10advices.com](#) (8) [www.srilankaholidayhotels.com](#) (9) [www.freebooknotes.com](#) (10) [danpritchard.com](#).

Top ten Web pages. (1) www.peoplesrepublicofcork.com/forums/showthread.php?p=3887808 (2) wapedia.mobi/en/List_of_countries_by_population (3) plumbot.com/List_of_countries_by_population.html (4) www.youramazingsmile.com/find.aspx?t=wiki&k=List_of_countries_by_population (5) en.wikipedia.org/wiki/List_of_countries_by_population (6) www.maxcaratulas.net/1/?title=List_of_countries_by_population (7) www.wiki.networkbase.info/read/List_of_countries_by_population (8) wiki.4worlds.net/en/List_of_countries_by_population (9) www.srilankaholidayhotels.com/wiki-List_of_countries_by_population (10) zomobo.com/Population-estimates-for-the-UK-June-2010.

A.15 D15

Task. Given a set of examples, e.g., {<George Washington, April 30, 1789, March 4, 1797, No party>, <Thomas Jefferson, March 4, 1801, March 4, 1809, Democratic-republican>}, the goal is to extract a list of instances of a 4-ary relation <US President, Date of Taking Office, Date of Leaving Office, Party>, i.e., a list of US presidents, the date they took office and left office, and parties they belonged to.

Top 20 candidate t-uples. (1) <george washington, april 30, 1789, march 4, 1797, no party> (2) <thomas jefferson, march 4, 1801, march 4, 1809, democratic-republican> (3) <bill clinton, january 20, 1993, january 20, 2001, democratic> (4) <andrew jackson, march 4, 1829, march 4, 1837, democratic> (5) <james madison, march 4, 1809, march 4, 1817, democratic-republican> (6) <woodrow wilson, march 4, 1913, march 4, 1921, democratic> (7) <richard nixon, january 20, 1969, august 9, 1974, republican> (8) <lyndon b. johnson, november 22, 1963, january 20, 1969, democratic> (9) <william henry harrison, march 4, 1841, april 4, 1841, whig> (10) <benjamin harrison, march 4, 1889, march 4, 1893, republican> (11) <james monroe, march 4, 1817, march 4, 1825, democratic-republican> (12) <grover cleveland, march 4, 1885, march 4, 1889, democratic> (13) <franklin pierce, march 4, 1853,

march 4, 1857, democratic> (14) <*theodore roosevelt, september 14, 1901, march 4, 1909, republican*> (15) <*dwight d. eisenhower, january 20, 1953, january 20, 1961, republican*> (16) <*gerald ford, august 9, 1974, january 20, 1977, republican*> (17) <*andrew johnson, april 15, 1865, march 4, 1869, democratic national union*> (18) <*ronald reagan, january 20, 1981, january 20, 1989, republican*> (19) <*william howard taft, march 4, 1909, march 4, 1913, republican*> (20) <*lyndon b. johnson, november 22, 1963, january 20, 1969, >*.

Top ten domains. (1) www.ask.com (2) wtfman.net (3) www.absoluteastronomy.com (4) www.enotes.com (5) answers.yahoo.com (6) mashpedia.com (7) pediaview.com (8) www.whmsoft.net (9) www.preceden.com (10) www.rq.sk.

Top ten Web pages. (1) www.telegraph.co.uk/news/worldnews/barackobama/4298481/The-43-Presidents-of-the-United-States-who-came-before-Barack-Obama.html (2) www.ask.com/wiki/List_of_Presidents_of_the_United_States (3) wapedia.mobi/en/List_of_Presidents_of_the_United_States (4) www.answers.com/topic/president-of-the-united-states-1 (5) www.archive.org/stream/cu31924079936153/cu31924079936153_djvu.txt (6) www.absoluteastronomy.com/topics/List_of_Presidents_of_the_United_States (7) www.cssforum.com.pk/css-optional-subjects/group-e-history-subjects/history-usa/15015-presidents-usa.html (8) www.infobarrel.com/A_List_of_United_States_Presidents (9) pdfcast.org/pdf/eighteenth-century-american-presidents (10) republicanscannotgovern.com/PresidentsoftheUnitedStatesofAmerica/PresidentsoftheUnitedStatesofAmerica.html.

