Language-Independent Class Instance Extraction Using the Web

Richard C. Wang

CMU-LTI-09-020

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213

Thesis Committee

William W. Cohen (Chair)
Robert E. Frederking
Tom M. Mitchell
Fernando C. N. Pereira (Google Research)

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Language and Information Technologies

Abstract

The World Wide Web is a vast and readily-available repository of factual information, such as semantic classes (e.g., fruits), their instances (e.g., orange, banana), and relations between them. There are many semi-structured documents on the web that provide evidence about these facts. The thesis of this work is that many of these facts can be revealed using tools built on set expansion. More generally, we believe that statistics, aggregation, and simple analysis of the documents are enough to discover frequent common classes in not only English, but other languages as well.

In this thesis, we formulate the discovery of semantic classes as *set expansion*, in which the user issues a query consisting of a small number of example instances (e.g., orange, banana) where each instance is a member of some target semantic class (e.g., fruits). The answer to the query is a listing of other probable instances of the class (e.g., apple, cherry).

We also illustrate that semi-structured documents provide more evidence and information than free text for discovering class instances. In addition, we propose a novel graph-based approach to set expansion that exploits semi-structured characteristics of web documents at character-level in a language-independent fashion. Furthermore, we develop a novel approach for our system to handle noisy instances, which allows it to be utilized as a tool for other applications, including question answering and WordNet extension. We show different techniques and strategies for enhancing the quality of expanded instances, including supervised, unsupervised, and bilingual methods. Lastly, we present an approach to perform relational set expansion using similar techniques.

This thesis is dedicated to my family for their love and support.

Acknowledgements

There are many people I would like to thank – this thesis could not be completed without their help. I would like to start by thanking my advisors William Cohen and Robert Frederking for their support, guidance, and freedom to work on this thesis. I would also like to thank my committee, Tom Mitchell and Fernando Pereira, for all the generous comments and support during my thesis. Furthermore, I would like to thank Eric Nyberg and Nico Schlaefer for their efforts during collaboration on the question answering paper. I would also like to thank Teruko Mitamura for the guidance on the development of the named entity translation system. Lastly, thanks to everyone who have tested my online system and provided me feedback.

Contents

List of Figures

List of Tables

LIST OF TABLES

Chapter 1

Introduction

1.1 Background

Automatic acquisition of class instances (e.g., president names, disease names, magazine names, and movie names) has been a well-studied problem. Comprehensive and accurate class-instance information has been proven useful in many applications, including relation learning from the web [5, 15, 31], feature generation for concept-learning [9] and co-reference resolution [23], dictionary construction for named entity recognition [2, 11, 25, 39, 41], query refinement in web search [26], enhancement of list-type answers [50] and document retrieval [32] for question answering, extensions to WordNet [38, 48], collaborative filtering [10], and similarity computation between attribute values in autonomous databases [52].

Class-instance information is contained in many textual documents, whether they are structured, semi-structured, or unstructured. Structured documents are text that conforms to fixed pre-determined schema; they are often found in databases. Semi-structured documents are texts that do not conform with the formal structure of tables and data models associated with databases but contain tags or other markers to separate semantic elements and hierarchies of records and fields within the text, such as HTML, XML, tab-separated, and comma-separated text. Unstructured documents (or free text) are text that have no markers and structure at all; such as newswire articles, conference publications, forum postings, web blogs, search engine query logs, and web pages stripped of HTML tags. Unlike structured documents, which often exist only in proprietary databases, many semi-structured documents and free text are readily-available on the web.

As a result, most research on class instance acquisition has been conducted using evidence from either semi-structured documents, free text, or a combination of both [8, 15, 42]. There are systems that utilizes semi-structured documents such as Wikipedia [38, 40, 54] and HTML web pages [6, 15, 25, 42]. There are many more systems that utilize unstructured free text such as *Grolier's American Academic Encyclopedia* [16, 17], *British National Corpus* [51], EachMovie corpus [16], the NIPS authors corpus [16], newswire articles from the MUC-4 [34], TREC-5 [38], and TREC-9 corpora [32], retrieved text snippets from search engines [20], web search query logs [27, 28], and the most popular repository – web pages stripped of HTML tags [8, 14, 15, 26, 29, 34, 38, 42].

Over the past few years, many research systems have been developed to extract class instances with various inputs. Some require only semantic class names (e.g., car makers) [15, 32], some require only example instances (e.g., Ford, Nissan, Toyota) [25, 27, 28, 34], and many require both [20, 38, 42, 51]. There are also systems that require no inputs at all but a handful of hand-crafted patterns [14, 17, 26]. However, all the abovementioned research systems have been shown to work only with documents written in English and not other languages; this is because they all rely on either capitalization, English part-of-speech taggers, English named entity recognizers, and/or English parsers.

For those research systems that utilize semi-structured documents, all of their methods focus only on the HTML structure and handle only HTML web pages (by using HTML renderers or parsers), which dramatically reduces their extraction recall, since they are ignoring many other semi-structured documents on the web such as XML and comma-separated documents, and also ignoring many useful hidden patterns occurring in web pages such as in URLs, file names, and HTML attributes.

1.2 Research Contributions

The World Wide Web is a vast and readily-available repository of factual information; such as semantic classes (e.g., fruits), their instances (e.g., orange, banana), and relations between them. There are many semi-structured documents on the web that provide evidence about these facts. The thesis of this work is that many of these facts can be revealed using tools built on set expansion. More generally, we believe that statistics, aggregation, and simple analysis of the documents are enough to discover frequent common classes in not only English, but other languages as well.

In this thesis, we formulate the discovery of semantic classes as *set expansion*, in which the user issues a query consisting of a small number of example instances x_1, x_2, ..., x_k (e.g., orange, banana) where each x_i is a member of some target set S (e.g., fruits). The answer to the query is a listing of other probable elements of S (e.g., apple, strawberry, and cherry). Google Sets[1] [44] is a well-known example of a web-based set expansion system.

Although Google Sets has been used for a number of purposes in the research community, including deriving features for named-entity recognition [37] and evaluation of question answering systems [33], unfortunately it is a proprietary system that may be changed at any time, so research results based on Google Sets cannot be reliably replicated. Hence, one contribution of this thesis is an open-source and documented version of Google Sets.

In this thesis, we developed a set expansion system that is based on our novel graph-based approach which exploits semi-structured characteristics of web documents at the character-level [46]. Through this system, we illustrate that semi-structured documents provide more evidence and information than free text for discovering class instances. Over the past few years, free text has remained as a popular common type of corpora for many instance extraction systems [14, 20, 28, 38, 41, 42]; however, we show that our approach achieves higher precision and recall than those free-text systems by utilizing semi-structured documents.

Unlike free text, semi-structured documents do not require language-specific preprocessing (e.g., parsing, tokenizing, sentences splitting, and part-of-speech tagging) to

[1] `http://labs.google.com/sets`

determine noun phrase boundaries; hence, we are able to make our approach independent of human language (e.g., English) of the documents. Since our approach utilizes character-level patterns, it is also independent of mark-up language (e.g., HTML) of the documents. We show that set expansion at the character-level performs better than at the HTML-level on semi-structured documents [49].

In addition, we illustrate that our set expansion system can be utilized as a tool for improving the accuracy of list-type answers from question answering systems [50] and for extending WordNet by using semantic class names as input [48]. However, when our system is being utilized as a tool by other automated systems, the instances input to our system are often noisy (i.e., containing mixture of relevant and irrelevant instances). To overcome this problem, we developed novel techniques for performing set expansion on noisy instances and show dramatic improvement in the quality of the expanded set, or *expansion quality*, over the noisy instances [48, 50].

Furthermore, we show different techniques and strategies for enhancing the expansion quality. We present supervised and unsupervised strategies to improve expansion quality by expanding instances iteratively [47]. We show that our unsupervised iterative (bootstrapping) approach is effective, and that our proposed graph-based ranking scheme is more robust to noisy instances than other state-of-the-art ranking schemes. We also present a novel technique to improve expansion quality by exploiting redundant information of classes in different languages. This is achieved by bootstrapping instances in two languages alternately, of which the instances in different languages are bridged together by a simple named entity translator.

Our set expansion approach was initially designed to extract only unary relations (e.g., x is a CEO). In this thesis, we present a variation of our approach to perform set expansion on binary relations (e.g., x is the CEO of company y). An example of input instances to our (binary) relational set expansion system is "*Bill Gates / Microsoft*" and "*Larry Page / Google*", of which our system produces other instance pairs of the same relation such as "*Larry Ellison / Oracle*". We show that our relational set expansion approach at character-level is effective and also performs better than HTML-based methods [49].

1.3 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 presents our set expansion system that uses our proposed character-level instance extraction and graph-based ranking approach. Chapter 3 presents our noise-resistant set expansion approach for expanding noisy instances, and shows that the approach improves question answering systems on list-type questions. Chapter 4 illustrates several iterative strategies to handle arbitrary number of input instances, and to improve expansion quality by bootstrapping. Chapter 5 describes an instance extraction system which takes as input a semantic class name and automatically outputs instances of the input class. Chapter 6 explains a technique for improving expansion quality by expanding instances in two languages alternately. Chapter 7 illustrates a relational set expansion approach that expands instance pairs with the same binary relation. Finally, related work is discussed in Chapter 8, and conclusions and future directions are presented in Chapter 9.

Chapter 2

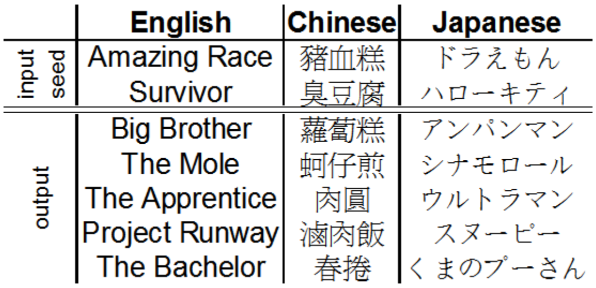
Set Expander for Any Language

2.1 Introduction

Have you ever wanted to find out the names of reality TV shows similar to the ones you regularly watch? In this chapter, we describe a set expansion system called the Set Expander for Any Language[1] (SEAL) that expands a given partial set of objects into a more complete set. It takes as inputs a few example seeds (e.g., Survivor) of a user-desired class (e.g., reality TV shows) and outputs more examples of that class (e.g., The Apprentice). More specifically, SEAL accepts input elements (seeds) of some target set S and automatically finds other probable elements of S in semi-structured documents such as web pages. SEAL also works on unstructured text (as described later in Section 2.3.3), but its extraction mechanism benefits from structuring elements (e.g., HTML tags). Unlike other published research work [15], SEAL focuses on finding small closed sets of instances (e.g., Disney movies) rather than large and more open sets (e.g., scientists). Examples of SEAL's input and output are shown in Figure 2.1.

As we will detail below, SEAL works by automatically finding semi-structured web pages that contain "lists" of candidate items, and then aggregating these "lists" so that the "most promising" items are ranked higher in the final output. SEAL's output is a ranked list because it is extremely difficult for machines to perfectly understand the information needs of users, and thus, to determine a perfect cut-off point in the ranked list. Unlike earlier work, the algorithm implemented in SEAL is simple enough to be easily described and replicated. It is independent of the human language (e.g., English,

[1] `http://rcwang.com/seal`



	English	Chinese	Japanese
input seed	Amazing Race	豬血糕	ドラえもん
	Survivor	臭豆腐	ハローキティ
output	Big Brother	蘿蔔糕	アンパンマン
	The Mole	蚵仔煎	シナモロール
	The Apprentice	肉圓	ウルトラマン
	Project Runway	滷肉飯	スヌーピー
	The Bachelor	春捲	くまのプーさん

Figure 2.1: Examples of SEAL's input and output. English instances are reality TV shows, Chinese are popular Taiwanese food, and Japanese are famous cartoon characters.

Chinese, Japanese) from which the seeds are taken, and is also independent of the markup language (e.g., HTML, XML, CSV) used to annotate the semi-structured documents. Extensive experiments have been conducted with SEAL, based on 36 benchmark problems from three languages, each of which consists of a moderate-sized set of instances that is semantically well-defined (e.g., constellations, major-league baseball teams). With randomly constructed three-seed queries from these domains, SEAL obtains a mean average precision (MAP) of more than 94% for English-language queries, more than 93% for Japanese queries, and nearly 95% for Chinese queries by using three seeds. MAP performance is more than double that of Google Sets.

In more detail, SEAL is based on two separate research contributions. To find "lists" of items on semi-structured pages, SEAL uses a novel technique to automatically construct wrappers (i.e., page-specific extraction rules) for each page that contains the seeds. Every wrapper is defined by two character strings, which specify the left-context and right-context necessary for an instance to be extracted from a page. These strings are chosen to be maximally-long contexts that bracket at least one occurrence of every seed string on a page. The use of character-level wrapper definitions means that SEAL is completely language-independent: it is not even necessary to be able to tokenize the target language.

Most of the wrappers that SEAL discovers will be "noisy" - i.e., they will extract some instances not in the user's target set. Thus, it is important to rank instances, so that the instances most likely to be in the target set are ranked higher. To rank instances, SEAL uses another novel approach: a graph is built containing all seeds, all constructed wrappers, and all extracted candidate instances. Candidates are then

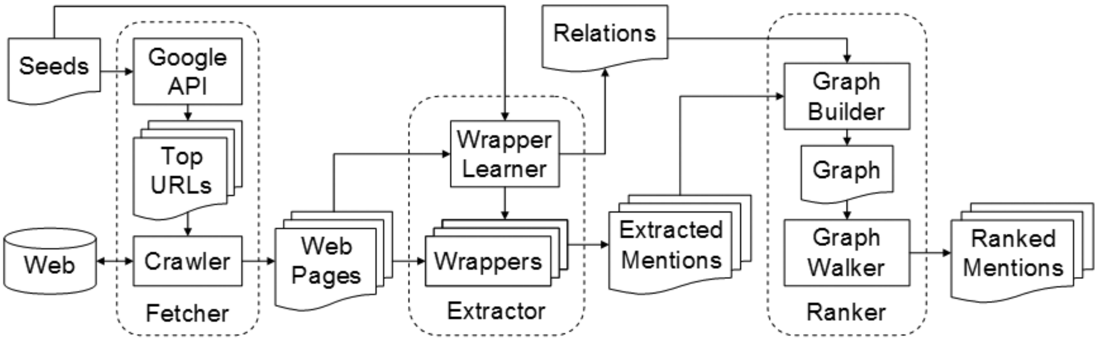


Figure 2.2: Flow chart of the SEAL system.

ranked by "similarity" to the seed instances, according to a certain measure of similarity in the graph. The similarity metric is defined by aggregating the results of many randomly-selected walks through the graph, where each walk is defined by a particular random process.

Information is organized in many different ways on the web. For instances that belong to the same class, we observed that they are frequently being placed within the same "list" on web pages. By extracting candidate instances that share the same contexts as the seeds, we are able to find those instances associated with the seeds by web users. We consider these instances to be similar to the seeds. We also observed that the more often an instance appears in the same list as the seeds, the more probable that the instance belongs to the same class as the seeds. By constructing a graph that models the relations between instances, wrappers, and documents, we are able to rank the instances based on how commonly they are being associated with the seeds, or how similar they are to the seeds.

In this chapter, Section 2.2 illustrates the architecture of SEAL. Section 2.3 describes how wrappers are constructed. We explain our ranking scheme based on graph walk in Section 2.4. Section 2.5 presents our evaluation methods and results, and we conduct qualitative comparisons to some previous work in Section 2.6. We summarize this chapter in the last section Section 2.7.

2.2 System Architecture

SEAL is comprised of three major components: the *Fetcher*, the *Extractor*, and the *Ranker*. The *Fetcher* is responsible for fetching web documents, and the URLs of the documents come from top results retrieved from Google using the concatenation of all seeds as the query (each seed is quoted, to require that it occur as an exact phrase). This ensures that every fetched web page contains all seeds.

The *Extractor* then learns and automatically constructs "wrappers" (i.e. page-specific extraction rules) for each page that contains the seeds. Every wrapper comprises two character strings that specify the left and right contexts necessary for extracting candidate instances. These contextual strings are maximally-long contexts that bracket at least one occurrence of every seed string on a page. All other candidate instances bracketed by these contextual strings derived from a particular page are extracted from the same page. The Extractor often extracts most of its candidate instances from semi-structured data pages found on the web. The character-level heuristics used to find wrappers often find candidate instances in parts of a web page that are not even easily visible to the end user (e.g., in pull-down menus of an HTML form).

After the instances are extracted, the *Ranker* constructs a graph that models all the relations between documents, wrappers, and candidate instances. The Ranker performs Random Walk with Restart [43] on this graph (where the initial "restart" set is the set of seeds) until all node weights converge, and then ranks candidate instances globally based on the weights computed in the graph walk; thus instances are weighted higher if they are connected to many seed nodes by many short, low fan-out paths. Figure 2.2 shows the architecture of SEAL. In this section, we describe the Extractor and Ranker in detail.

2.3 The Extractor

The extractor must learn wrappers instantly and automatically from only a few training examples (the seeds). In this section, we explain the semi-structured characteristics of web documents that SEAL requires, and describe an unsupervised approach for automatic construction of wrappers. The wrappers that are constructed are page-dependent - i.e., they are intended to be applied only to a single web page. However, the approach that we use to learn wrappers is both domain- and language- independent.

2.3.1 Semi-Structured Documents

We assume information in semi-structured web documents will be formatted quite differently on different documents, but fairly consistently within a single document. For example, each movie name in a HTML web page containing a list of Disney movies might be embedded with "`<tr><td>`" (to the left) and "`</td></tr>`" (to the right) in one page, and "`<ul>Disney:`" and "`</ul>`" in another. This observation suggests that items belonging to the same class (i.e. movies) will be linked by appearing in similar contexts (formatting structures) on the same document. This occurs not only in HTML documents, but also in other semi-structured documents such as XML (.xml), comma-separated (.csv), tab-separated (.tsv), latex (.tex), etc.

This characteristic of semi-structured web documents can be exploited for expanding some set of given seeds. Suppose initially, a couple of seeds are provided from a reliable source (i.e., a human), and documents that contain all of these seeds are retrieved. Then it is very likely that each of these documents will contain other instances that are embedded in the same contexts as the seeds, and also belong to the same semantic class as the seeds. The section below explains in detail the algorithm for constructing wrappers utilizing the semi-structured characteristics of semi-structured documents.

2.3.2 Identifying Wrappers for Unary Relations

When SEAL performs set expansion, it accepts a small number of *seeds* from the user (e.g., "*Ford*", "*Nissan*", and "*Toyota*"). It then uses a web search engine to retrieve some documents that contain these instances, and then analyzes these documents to find *candidate wrappers* (i.e., regular structures on a page that contain the seed instances). Strings that are extracted by a candidate wrapper (but are not equivalent to any seed) are called *candidate instances*. SEAL then statistically ranks the candidate instances (and wrappers), using the techniques outlined below, and outputs a ranked list of instances to the user.

One key step in this process is identifying candidate wrappers. In SEAL, a candidate wrapper is defined by a pair of left and right character strings, ℓ and r. A wrapper "extracts" items from a particular document by locating all strings in the document that are bracketed by the wrapper's left and right strings, but do not contain either of

URL:	http://www.shopcarparts.com/
Wrapper:	.html" CLASS="shopcp">[...] Parts</A>
Content:	acura, audi, bmw, buick, cadillac, chevrolet, chevy, chrysler, daewoo, daihatsu, dodge, eagle, ford, ...
URL:	http://www.allautoreviews.com/
Wrapper:	</a> <a href="auto_reviews/[...]/
Content:	acura, audi, bmw, buick, cadillac, chevrolet, chrysler, dodge, ford, gmc, honda, hyundai, infiniti, isuzu, ...
URL:	http://www.hertrichs.com/
Wrapper:	<li class="franchise [...]"> <h4><a href="\#">
Content:	buick, chevrolet, chrysler, dodge, ford, gmc, isuzu, jeep, lincoln, mazda, mercury, nissan, pontiac, scion, ...
URL:	http://www.metacafe.com/watch/1872759/2009_nissan_maxima_performance/
Wrapper:	videos">[...]</a> <a href="/tags/
Content:	avalon, cars, carscom, driving, ford, maxima, nissan, performance, speed, toyota
URL:	http://www.worldstyling.com/
Wrapper:	'>[...] Accessories</option><option value='
Content:	chevy, ford, isuzu, mitsubishi, nissan, pickup, stainless steel, suv, toyota

Table 2.1: Examples of wrappers constructed from web pages and their extracted contents (candidate instances) given the seeds: *Ford*, *Nissan*, and *Toyota*.

the two strings. In SEAL, wrappers are always learned from, and applied to, a single document.

Table 2.1 illustrates some candidate wrappers learned by SEAL. (Here, a wrapper is written as ℓ[...]r, with the [...] to be filled by an extracted string.) Notice that the instances extracted by wrappers can and do appear in surprising places, such as embedded in URLs or in HTML tag attributes. Our experience with these character-based wrappers lead us to conjecture that existing heuristics for identifying structure in HTML are fundamentally limited, in that many potentially useful structures will not be identified by analyzing HTML structure only. We also conjecture that the less constrained character-level methods will produce more candidate wrappers than HTML-based techniques, and a larger number of candidate wrappers will lead to better performance overall. We investigated our conjectures by using page analysis techniques constrained to identify only HTML-related wrappers in Section 2.5 and present experimental results in Table 2.10.

SEAL uses the following rule to find wrappers. Each candidate wrapper ℓ, r is a maximally long pair of strings that bracket at least one occurrence of every seed in a document: in other words, for each pair ℓ, r, the set of strings C extracted by ℓ, r has the properties that:

1. For every seed s, there exists some $c \in C$ that is equivalent to s; and
2. There are no strings ℓ', r' that satisfy property (1) above such that ℓ is a proper suffix of ℓ' and r is a proper prefix of r'.

SEAL's wrappers can be found quite efficiently using the algorithm described in Table 2.2. As an example, below shows a mock document, written in an unknown markup language, that has the seeds: Ford, Nissan, and Toyota located (and underlined). There are two other car makers hidden inside this document. In this section, we show how to automatically construct wrappers that reveal them.

```
GtpKxHNissanxjHJglekuDialcLBxKHFordxkrpW
NaCMwAAHOForduohdEXocUvaGKxHAcuraxjHjnOx
oToyotazxKHAudixkrOyQKxHToyotaxjHCRdmLxa
puRAPprtqOVKxHFordxjHaJAScRFrlaFordofwNL
WxKHToyotaxkrHxQKlacXlGEKtxKHNissanxkrEq
```

Given a set of seeds and a semi-structured document, the wrapper construction algorithm starts by locating all strings equivalent to a seed in the document; these strings are called *seed instances* below. (In SEAL, we always use case-insensitive string matching, so a string is "equivalent to" any case variant of itself.) The algorithm then inserts all the instances into a list and assigns a unique *id* to each of them (the *id* of an instance is its position in the list.)

For every seed instance in the document, its immediate left context (starting from the first character of the document) and right context (ending at the last character of the document) are extracted and inserted into a *left-context* trie and a *right-context* trie respectively, where the left context is inserted in reversed character order. Here, we implemented a compact trie called a Patricia trie [24], which is a specialized set data structure based on the trie that is used to store a set of strings. Unlike a regular trie, the edges of a Patricia trie are labelled with sequences of characters rather than with single characters. Every node in the left-context trie maintains a list of *id*s for keeping track of the seed instances that follow the string associated with that node. Same thing

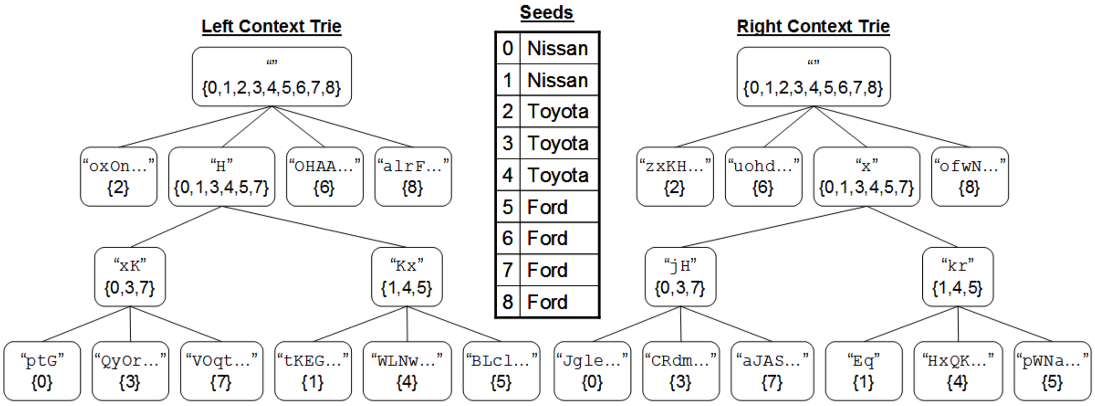


Figure 2.3: The context tries and the seed instance list constructed given the mock document and the seeds: Ford, Nissan and Toyota.

applies to the right-context trie symmetrically. Figure 2.3 shows the two context tries and the list of seed instances when provided the mock document with the seeds: Ford, Nissan, and Toyota.

Provided that the left and right context tries are populated with all the contextual strings of every seed instance, the algorithm then finds maximally long contextual strings that bracket at least one seed instance of every seed. The pseudo-code for finding these strings for building wrappers is illustrated in Table 2.2, where *Seeds* is the set of input seeds and ℓ is the minimum length of the strings. We observed that usually higher values of ℓ produce higher precision but lower recall. This is an interesting parameter that is worth exploring, but for this thesis, we use a value of one throughout the experiments. The basic idea behind the pseudo-code is to first find all the longest possible strings from one trie given some constraints, then for every such string s, find the longest possible string s' from another trie such that s and s' bracket at least one occurrence of every given seed in a document.

The wrappers constructed as well as the items extracted given the mock document and the example seeds are shown below. Notice that *Audi* and *Acura* are discovered.

Wrapper:	`xKH[...]xkr`
Content:	**Audi**, Ford, Nissan, Toyota
Wrapper:	`KxH[...]xjH`
Content:	**Acura**, Ford, Nissan, Toyota

```
**Wrappers MakeWrappers(Trie $\ell$, Trie $r$)**
  Return Wraps($l, r$) $\cup$ Wraps($r, l$)

Wrappers Wraps(Trie $t_1$, Trie $t_2$)
  For each $n_1 \in$ TopNodes($t_1$, $\ell$)
    For each $n_2 \in$ BottomNodes($t_2$, $n_1$)
      For each $n_1 \in$ BottomNodes($t_1$, $n_2$)
        Construct a new Wrapper(Text($n_1$), 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) NumCommonSeeds($n$, $n'$) $==$ $|Seeds|$, and
    (2) All children nodes of $n$ (if exist) fail on (1)
  Return a union of all nodes found

Nodes TopNodes(Trie $t$, int $\ell$)
  Find node $n \in t$ such that:
    (1) 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 $i \in$ 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 $n_1.indexes \cap n_2.indexes$
```

Table 2.2: Pseudo-code for constructing wrappers.

Table 2.1 shows real examples of wrappers constructed from some randomly selected web pages given the seeds: *Ford*, *Nissan*, and *Toyota*. Examining more closely at one particular web page, Table 2.3 shows a portion of the HTML source code of the web page from `http://www.curryauto.com` with the seeds underlined, and Table 2.4 shows some wrappers constructed as well as the candidate instances they extract from the page. We have also observed instances extracted from plain text (.txt), comma/tab-separated text (.csv/.tsv), latex (.tex), and even Word documents (.doc) for which the wrappers have binary character strings. These observations support our claim that the algorithm is independent of mark-up language. In our experimental results, we will show that it is independent of human language as well.

2.3.3 Free-Text Wrappers

Unstructured documents (or free text) are text that have no markers and structure at all; such as newswire articles, conference publications, forum postings, web blogs, search engine query logs, and web pages stripped of HTML tags. Although our extraction mechanism benefits from structuring elements (e.g., HTML tags), we have often observed that the contexts in our constructed wrappers contain only free text as well, and most importantly, these wrappers extracted many relevant instances. Below, we show some examples of free-text wrappers constructed by our approach given *Ford*, *Nissan*, and *Toyota* as seeds.

```
“Used [...] Van Pricing”
“Used [...] Engines”
“Bell Road [...] ”
“Alaska [...] dealership”
“www.sunnyking[...].com"”
“engine [...] used engines”
“accessories, [...] parts”
“is better [...] or”
```

We estimated that these free-text wrappers consist of 10-20% of all wrappers constructed by our proposed approach. This observation shows that there are meaningful patterns hidden in web pages that are usually being ignored by systems that utilize only

```
<li class="ford"><a href="http://www.curryford.com/">
<img src="/common/logos/ford/logo-horiz-rgb-lg-dkbg.gif" alt="3"></a>
   <ul><li class="last"><a href="http://www.curryford.com/">
      <span class="dName">Curry Ford</span>...</li></ul>
</li>
<li class="honda"><a href="http://www.curryhonda.com">
<img src="/common/logos/honda/logo-horiz-rgb-lg-dkbg.gif" alt="4"></a>
   <ul><li><a href="http://www.curryhonda-ga.com/">
      <span class="dName">Curry Honda Atlanta</span>...</li>
      <li><a href="http://www.curryhondamass.com/">
         <span class="dName">Curry Honda</span>...</li>
      <li class="last"><a href="http://www.curryhondany.com/">
         <span class="dName">Curry Honda Yorktown</span>...</li></ul>
</li>
<li class="acura"><a href="http://www.curryacura.com/">
<img src="/curryautogroup/images/logo-horiz-rgb-lg-dkbg.gif" alt="5"></a>
   <ul><li class="last"><a href="http://www.curryacura.com/">
      <span class="dName">Curry Acura</span>...</li></ul>
</li>
<li class="nissan"><a href="http://www.geisauto.com/nissan/">
<img src="/common/logos/nissan/logo-horiz-rgb-lg-dkbg.gif" alt="6"></a>
   <ul><li class="last"><a href="http://www.geisauto.com/nissan/">
      <span class="dName">Curry Nissan</span>...</li></ul>
</li>
<li class="toyota"><a href="http://www.geisauto.com/toyota/">
<img src="/common/logos/toyota/logo-horiz-rgb-lg-dkbg.gif" alt="7"></a>
   <ul><li class="last"><a href="http://www.geisauto.com/toyota/">
      <span class="dName">Curry Toyota</span>...</li></ul>
</li>
```

Table 2.3: A portion of HTML source code from `http://www.curryauto.com`

Wrapper:	`\n<li class="[...]"><a href="http://www.`
Content:	ford, honda, acura, kia, toyota, scion, nissan, buick, pontiac
Wrapper:	`/">\n<img src="/common/logos/[...]/logo-horiz-rgb-lg-dkbg.gif" alt="`
Content:	chevrolet, ford, kia, toyota, scion, nissan, pontiac, cadillac, hyundai
Wrapper:	`<span class="dName">Curry [...]</span>`
Content:	chevrolet, ford, honda atlanta, honda, honda yorktown, acura, subaru chicopee, subaru, kia, toyota, scion, nissan, buick, pontiac, cadillac

Table 2.4: Wrappers induced from `http://www.curryauto.com` and their extracted contents (candidate instances) given the seeds: *Ford*, *Nissan*, and *Toyota*.

HTML tags on web pages [6, 15, 25, 42]. In addition, it also shows that our approach is capable of constructing wrappers from unstructured documents as well.

2.4 The Ranker

The set of candidates extracted by wrappers may contain noisy instances (i.e., instances that are rarely associated with the seeds by popular consensus.) For example, *Honda Atlanta* and *Honda Yorktown* extracted by the third wrapper in Table 2.4 are such instances; these are unlikely to be members of the user's target set. Since it is extremely difficult for machines to perfectly understand the information needs of users, we choose to rank the candidate instances in the set presented to the users. In this section, we first analyze the problem of finding similarity between seeds and candidate instances, then we propose a graph walk method for ranking those instances.

2.4.1 Analyzing the Problem

In order to determine the similarity between candidate instances and seeds (or the likelihood that they all belong to the same class based on contextual information), we need to first understand how they are related globally. We know that seeds were used as a query to find documents online, and the same instance can be extracted by more than one wrapper. Also, we have observed that noisy instances are usually extracted less frequently than non-noisy instances. Intuitively, the more non-noisy instances extracted by a wrapper, the better quality the wrapper (and vice versa), and the more high-quality wrappers derived from a document, the better quality the document (and vice versa). In order to model these complex relations, we will use a graph which encapsulates the relations between all objects of interest - web documents, wrappers, and candidate instances. Similarity in the graph will then be used to rank candidate instances.

2.4.2 Building a Graph

A graph G consists of a set of nodes and a set of labeled directed edges. Figure 2.4 shows an example graph where each node d_i represents a document, w_i a wrapper, and m_i a candidate instance. A directed edge connects a node d_i to a w_i if d_i contains w_i, a w_i to a m_i if w_i extracts m_i, and a d_i to a m_i if d_i contains m_i. Although not shown

Source Type	Edge Relation	Target Type
document	contains	wrapper
document	contains	instance
wrapper	extracts	instance
wrapper	is contained by	document
instance	is contained by	document
instance	is extracted by	wrapper

Table 2.5: All possible source node types and their relations with some target node types.

in the figure, every edge from node x to y actually has an inverse relation edge from node y to x (e.g., m_i is extracted by w_i) to ensure that the graph is cyclic. Table 2.5 shows all possible source node types and their relations with some target node types.

In order to explain the random walk equations in detail, we will use letters such as x, y, and z to denote nodes, and $x \xrightarrow{r} y$ to denote an edge from x to y with labeled relation r. Each node represents an object (document, wrapper, or instance), and each edge $x \xrightarrow{r} y$ asserts that a binary relation $r(x, y)$ holds.

2.4.3 Random Walk with Restart

We want to find candidate instance nodes that are *similar* to the seed nodes. We define the similarity between two nodes by random walk with restart [43]. The general idea, however, is that nodes should be weighted higher if they are connected to many other highly weighted (important) nodes. In this algorithm, to walk away from a source node x, one first chooses an edge relation r; then given r, one picks a target node y such that $x \xrightarrow{r} y$. When given a source node x, we assume that the probability of picking an edge relation r is uniformly distributed among the set of all r, where there exist a target node y such that $x \xrightarrow{r} y$. More specifically,

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

We also assume that once an edge relation r is chosen, a target node y is picked uniformly from the set of all y such that $x \xrightarrow{r} y$. More specifically,

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

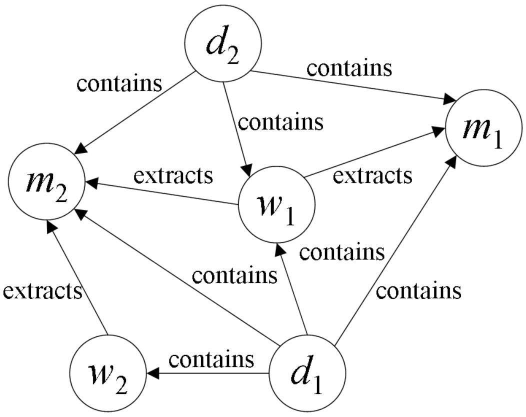


Figure 2.4: Example graph built by Random Walk. Notice that every edge from node x to y actually has an inverse relation edge from node y to x (e.g., m_i is extracted by w_i).

In order to perform random walk, we will build a transition matrix M where each entry at (x, y) represents the probability of traveling one step from a source node x to a target node y, or more specifically,

$$M_{xy} = \sum_r P(r|x)P(y|r, x)$$

We will also define a state vector $\vec{v_t}$ which represents the probability at each node after iterating through the entire graph t times, where one iteration means to walk one step away from every node. The state vector at $t + 1$ iteration is defined as:

$$\vec{v}_{t+1} = \lambda\vec{v}_0 + (1 - \lambda)M\vec{v}_t$$

Since we want to start our walk from the seeds, we initialize v_0 to have probabilities uniformly distributed over the seed nodes. In each step of our walk, there is a small probability λ of teleporting back to the seed nodes, which prevents us from walking too far away from the seeds. We iterate our graph until the state vector converges, and rank the candidate instances by their probabilities in the final state vector. The final expanded set contains all candidate instances in the graph, ranked globally by their weights in the graph. In the experiments below, we use a constant λ of 0.01, which shows good performance in our development set. New statistics can be accumulated easily by adding additional nodes and edges to the existing graph.

2.5 Evaluation

In this section, we describe a baseline system, alternative methods we attempted to use, evaluation datasets, evaluation methods, and evaluation results.

2.5.1 Baseline System

We choose Google Sets[1] [44] as our baseline system, mainly because it is a well-known example of a web-based set expansion system that is publicly available. Google Sets has been used for numerous purposes, including deriving features for named entity recognition [37] and evaluation of question answering systems [33]. However, it is a proprietary method that may be changed at any time, so research results based on Google Sets cannot be reliably replicated.

Google Sets contains a list identifier, a list classifier and a list processor. The list identifier identifies existing lists by a HTML tag (e.g., `<UL>`, `<OL>`, `<DL>`, `<H1>-<H6>` tags), by items placed in a table, items separated by commas or semicolons, or items separated by tabs. The list classifier generates an on-topic model and determine confidence scores that the existing lists were generated using the on-topic model. The list processor forms a list from the items in the existing lists and the determined confidence scores associated with the existing lists.

2.5.2 Alternative Rankers

We conducted ablation studies using several alternative rankers for set expansion. In this section, we present three alternative rankers used in our experiments, namely PageRank (PR), Bayesian Sets (BS), and Wrapper Length (WL). A fourth ranker used in our experiments is called Wrapper Frequency (WF), which simply ranks each candidate instance i by the number of wrappers that extract i.

2.5.2.1 PageRank

Page *et al.* [30] proposed the PageRank algorithm that is being used extensively at Google to score web pages. Although it was designed to rank hyperlinked set of documents (i.e., web pages), many research have shown that it can also be used to rank other things [45, 53]. PageRank measures relative importance of each node within a

[1] `http://labs.google.com/sets`

graph; it interprets a link from node A to node B as a vote, by node A, for node B. However, the weights of the votes are not the same. Votes cast by nodes that are themselves *important* weigh more heavily and help to make other nodes *important*.

The graph that we use for PageRank is identical to the one shown in Figure 2.4, except that edges are undirected and they do not have relations. In order to compute PageRank scores, we will also build a transition matrix M where each entry at (x, y) is the probability of traveling one step from a source node x to a target node y, or more precisely,

$$M_{xy} = \frac{1}{|y : x \longrightarrow y|}$$

We also define a state vector $\vec{v_t}$, and the state vector at $t+1$ iteration is defined as:

$$\vec{v}_{t+1} = \lambda\vec{u} + (1-\lambda)M\vec{v}_t$$

where $\vec{u}$ is a teleport vector with probabilities uniformly distributed over all nodes in the graph. Unlike random walk, each node in PageRank has an uniform probability λ of teleporting to an arbitrary node in the graph; thus, v_0 is not relevant. Page *et al.* [30] uses a λ of 0.15, which is also what we use in the experiments. We iterate the graph until the state vector converges, and rank the candidate instances based on their final node weights. Note that new statistics can be easily accumulated by attaching new nodes and edges to the existing graph.

2.5.2.2 Bayesian Sets

Ghahramani and Heller [16] proposed a ranking algorithm called the Bayesian Sets, which formulates the set expansion problem as a Bayesian inference problem. It uses a model-based concept of a class and ranks items using a score which evaluates the marginal probability that each item belongs to the class containing the seed items. More specifically, having observed a set of seed items S belonging to some concept, they measure the probability that an item i also belongs with S by $p(i|S)$. However, ranking items simply by this probability is not sensible because some items may be more probable than others, regardless of S. To overcome this problem, they compute the ratio:

$$score(i) = \frac{p(i|S)}{p(i)} = \frac{p(i,S)}{p(i)p(S)}$$

which can be interpreted as the ratio of the joint probability of observing i and S, to the probability of independently observing i and S. In order to compute the above equation, they assume that the data points in the cluster all come independently and identically distributed from a simple parameterized statistical model: $p(i|\theta)$. They assume each item i has an independent Bernoulli distribution, and they introduce two hyperparameters α and β for the Beta distribution: $p(\theta|\alpha, \beta)$, which is the conjugate prior for the parameters of a Bernoulli distribution. For all the assumptions they made above, they show that there exists a closed-form solution.

We used Bayesian Sets by constructing one large two-dimensional feature table where each column represents a particular feature, each row represents a particular extracted item, and each entry (j, k) in the table indicates whether item i_k possesses the feature f_j. We incorporate two important features: document containment and wrapper extraction. For example, if an item is contained by a document d_j or was extracted by a wrapper w_j, then entry (j, k) would be 1; otherwise 0. We tried using either one of the features alone on our development set, but the results are worse. For an extracted item i_k, we calculate its score using the closed-form solution described in their work:

$$\log\ score(i_k) = \mathbf{C} + \sum_{j:(j,k)=1} \Big[\log(\alpha_j + n_j) - \log(cm_j) - \log(\beta_j + N - n_j) + \log(c(1 - m_j))\Big]$$

where $\mathbf{C}$ is some constant, m_j is the mean value of column j, n_j is the number of seeds contained by d_j or extracted by w_j, N is the total number of seeds, and c is a parameter which is set to 2, as also done by Ghahramani and Heller [16]. The basic concept is that an item will have a higher score if it is a) extracted by many wrappers that extract few items and b) extracted from many documents that contain few extracted items. Note that new statistics can be accumulated with the existing ones simply by appending new rows and columns to the feature table.

2.5.2.3 Wrapper Length

SEAL defines a wrapper, which extracts item i, as the maximally-long contextual strings that i has in common with the input seeds in a document. We have observed that the longer the common left and right contextual strings, the more likely that i is correlated with the seeds in that document. Therefore, we propose a simple, fast but ad hoc ranking algorithm called Wrapper Length as detailed below:

$$\log score(i) = \sum_{j \text{ extracts } i} \frac{\log(length(w_j))}{\log(length(d_j))}$$

where w_j is the j^{th} wrapper composed of a pair of left and right contextual strings, d_j is the document of which w_j is derived from, and the function $length(s)$ returns the character length of s (i.e., the total number of characters in a document or in the pair of strings of a wrapper). This heuristic is based on the assumption that an item should have a high score if it is extracted by many long wrappers. In this approach, new statistics can be accumulated simply by summing up the quotient of the log of wrapper and document lengths for every new item, and the memory size needed to store these statistics is only proportional to the number of items.

2.5.3 Alternative Extractors

We also conducted ablation studies using several alternative extractors for set expansion. In this section, we describe one simple extractor and four types of HTML-based wrappers. The simple extractor, referred to as SE, finds maximally-long contextual strings that bracket *all* seed occurrences, instead of bracketing *at least one* occurrence of every seed. This simplified extractor is compared to our proposed extractor, referred to as PE, in the evaluation results section.

PE builds character-based wrappers that do not have any restriction on the alphabets of their characters. We would like to determine whether character-based or HTML-based wrappers are more suited for the task of set expansion. SEAL currently does not use an HTML parser (or any other kinds of parser), so additional HTML restrictions cannot be easily imposed. As far as we know, there is not an agreement on what restrictions make the most sense or work the best. Therefore, we evaluate performance by varying wrapper constraints from type 1 (less strict) to type 4 (more strict) in our experiments. In order to do that, we introduce four types of HTML-based wrappers, referred to as H1 to H4. The first type (H1) requires that each of the left and right contextual strings must contain either < or >. The fourth type (H4) requires that an item must be tightly bracketed by two complete HTML tags in order to be extracted. From H1 to H4, the allowable alphabets in a wrapper become more and more restrictive as illustrated in Table 2.6, where H2 and H3 are intermediate points. Note

	Left [...] Right
PE	.+[...].+
H1	.*[<>].*[...].*[<>].*
H2	.*>[...]<.*
H3	.*<.+?>.*[...].*<.+?>.*
H4	.*<.+?>[...]<.+?>.*

Table 2.6: Regular expressions illustrating character restrictions for our proposed extractor (PE) and the four types of HTML-based wrappers (H1-H4).

that all pure HTML-based wrappers are type 4, possibly with additional restrictions imposed [15, 25].

2.5.4 Evaluation Datasets

We manually constructed 36 evaluation datasets evenly across three different languages: English, Chinese, and Japanese; thus there are 12 datasets per language. The datasets consist of 18 semantic classes, where half were constructed in all three languages and the other half in one language only, as illustrated in Table 2.7. The intention is to diversify the datasets such that some are culture-specific while some are not. A full detailed list of instances contained in each of the datasets is presented in Appendix A. Each dataset is a true list of a particular semantic class C, and each instance $i \in C$ is represented by a list of synonyms (e.g., USA, America) of that particular instance i. The statistics of semantic classes, instances, and synonyms for each language are shown in Table 2.8.

2.5.5 Evaluation Method

Since the output of our system is a ranked list of extracted instances, we choose mean average precision (MAP) as our evaluation metric. MAP is commonly used in the field of Information Retrieval for evaluating ranked lists because it is sensitive to the entire ranking and it contains both recall and precision-oriented aspects. The MAP for multiple ranked lists is simply the mean value of average precisions calculated separately for each ranked list. We define the average precision of a single ranked list as:

	Dataset ID	Eng	Chi	Jap	Dataset Description
1	disney-movies	√	√	√	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	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 building names
11	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-prime-mins			√	Japanese prime minister names
18	japan-provinces			√	Japanese province names

Table 2.7: The 36 evaluation datasets; 12 in each of the following languages: English, Chinese and Japanese.

Language	Class(C)	Instance(I)	Synonym(S)	Avg. I/C	Avg. S/I
English	12	1028	1500	86	1.5
Chinese	12	701	1744	58	2.5
Japanese	12	809	1696	67	2.1

Table 2.8: Statistics of the 36 evaluation datasets, which includes the number of semantic classes, instances, and synonyms, as well as the average number of instances per class and average number of synonyms per instance.

$$AvgPrec(L) = \frac{\sum_{r=1}^{|L|} \text{Prec}(r) \times \text{isFresh}(r)}{\text{Total \# of Correct Instances}}$$

where L is a ranked list of extracted instances, r is the rank ranging from 1 to $|L|$, Prec(r) is the precision at rank r, or the percentage of correct synonyms above rank r (inclusively). The binary function isFresh(r) ensures that, if a list contains multiple synonyms of the same instance, we do not evaluate that instance more than once. More specifically, the function returns 1 if a) the synonym at r is correct, and b) it is the highest-ranked synonym of its instance in the list; it returns 0 otherwise. Note that the denominator of the equation is actually the total number of correct instances in the evaluation dataset; otherwise, there is no sense of recall.

We evaluated SEAL by giving it some seeds randomly selected from the evaluation datasets and then determine the quality of its output lists. More specifically, the procedure for evaluating SEAL is, for each dataset:

1. Randomly select three instances and use their first listed synonym[1] as seeds.
2. Expand the three seeds obtained from step 1.
3. Repeat steps 1 and 2 three times.
4. Compute MAP for the three resulting ranked lists.

Notice that there are several parameters that can be tuned to improve the recall and precision of SEAL. For example, the recall could be improved by requesting more web pages from the search engines at the expense of longer processing time. In addition, with some sacrifice of recall, the precision could be improved by, for example, a) requiring each left and right context of a wrapper to contain certain HTML tags, or b) requiring each instance to be extracted by more than one wrapper. In all the experiments we conducted in this thesis (unless otherwise specified), we did not enforce any restriction on the contexts of the wrappers, and we also did not require each instance to be extracted by a certain number of wrappers. However, in order to minimize runtime, we did configure SEAL to retrieve only one hundred web pages.

2.5.6 Evaluation Results

Table 2.9 shows our experimental results using three seeds. Here, we refer to our extractor proposed in Section 2.3.2 as PE and our random walk approach proposed in Section 2.4.3 as RW. As shown, the baseline Google Sets (G.Sets) performed the worst; even our simplest approach, Simple Extractor (SE) plus Wrapper Frequency (WF), beats Google Sets by a substantial amount. When we requested only top 100 URLs per expansion from Google, our simplest approach (SE+WF) achieved an overall average of around 82%. After replacing SE with our proposed extractor PE, the overall

[1] Most commonly-known name of the instance (e.g., “Bill Clinton” vs. “William Jefferson Clinton”).

average improved to about 88% (a 6.3% improvement). When WF is replaced with our proposed ranker RW, the overall average improved to about 93% (another 6.3% improvement). We wanted to know whether the corpus size has any effect on the set expansion performance. Therefore, we increased the number of web pages from 100 to 200 and 300, and we observed a slight improvement of 1.0% and 0.2% respectively (from 93% MAP to 94.0% and 94.2% respectively). In brief, this table shows that our proposed extractor PE is more effective than SE and that our proposed ranker RW is more effective than WF. In addition, the results also show that the more web pages, the better the quality of the expanded set of items.

In the first set of experiments, we have shown that PE is an effective extractor. In the next set of experiments, we evaluated various alternative rankers as presented in Section 2.5.2 with the extractor PE. To make the set expansion problem more challenging and accentuate differences between systems, we used only *two* seeds instead of *three*. The experimental results are illustrated in Figure 2.5. As shown, no matter how many web pages were being used, our proposed ranker RW is always superior than other rankers, followed closely by Bayesian Sets (BS) and Wrapper Length (WL). The ranker that has the worse performance is PageRank (PR). Note that BS did not perform well when the corpus size is small. Again, these results illustrated that the more web pages, the better the set expansion results.

In the previous two sets of experiments, we have shown that RW is an effective ranker. In the next set of experiments, we evaluated various alternative extractors as presented in Section 2.5.3 with the ranker RW. Again, we evaluated using only two seeds. From previous experiments, we observed that a maximum of 200 web pages gives a good balance between performance and speed; therefore, we used a maximum of 200 web pages in this set of experiments. The experimental results are presented in Table 2.10. As shown, the more restrictive the wrappers, the worse the performance. Our proposed approach (PE+RW) has the best performance because PE impose no restriction on the wrappers. As a comparison, we included the performance of Google Sets (G.Sets) using two seeds, which performed the worse, again.

		Max. 100 Pages			Max. 200	Max. 300
English	**G.Sets**	**SE+WF**	**PE+WF**	**PE+RW**	**PE+RW**	**PE+RW**
disney-movies	46.5	79.4	74.5	84.4	88.2	89.4
constellations	49.8	89.6	100.0	100.0	100.0	100.0
countries	22.3	96.0	97.9	98.2	98.7	98.5
mlb-teams	97.7	98.6	99.5	99.8	99.8	99.8
nba-teams	100.0	100.0	100.0	100.0	100.0	100.0
nfl-teams	100.0	99.2	100.0	100.0	100.0	100.0
car-makers	57.8	79.2	88.2	95.2	96.2	97.0
us-presidents	99.9	91.6	97.1	100.0	100.0	100.0
us-states	81.6	100.0	93.5	100.0	100.0	100.0
cmu-buildings	0.0	87.8	87.8	87.8	87.8	87.8
diseases	5.1	17.9	52.8	57.5	75.8	76.9
periodic-comets	0.0	69.2	79.0	84.8	84.8	84.8
Average	**55.1**	**84.0**	**89.2**	**92.3**	**94.3**	**94.5**
Chinese	**G.Sets**	**SE+WF**	**PE+WF**	**PE+RW**	**PE+RW**	**PE+RW**
disney-movies	3.3	80.7	91.2	91.7	91.7	91.7
constellations	22.8	92.0	96.3	100.0	100.0	100.0
countries	8.4	94.8	95.4	96.9	97.8	97.7
mlb-teams	59.2	94.4	84.0	100.0	100.0	100.0
nba-teams	0.0	90.3	95.0	99.9	100.0	100.0
nfl-teams	0.0	68.1	88.4	95.7	95.7	95.7
car-makers	4.8	71.4	83.3	94.4	94.5	94.6
us-presidents	0.0	62.8	82.6	93.0	94.2	94.2
us-states	6.6	98.5	97.1	99.5	99.5	99.5
china-dynasties	4.5	25.4	33.9	65.2	64.6	65.2
china-provinces	87.9	95.0	99.2	99.2	99.3	99.4
taiwan-cities	100.0	95.3	98.0	100.0	100.0	100.0
Average	**24.8**	**80.7**	**87.0**	**94.6**	**94.8**	**94.8**
Japanese	**G.Sets**	**SE+WF**	**PE+WF**	**PE+RW**	**PE+RW**	**PE+RW**
disney-movies	7.6	72.8	75.0	81.6	83.0	83.0
constellations	38.6	96.9	95.2	100.0	100.0	100.0
countries	24.0	97.3	90.5	98.7	99.2	99.2
mlb-teams	0.0	80.0	85.6	98.9	98.9	98.9
nba-teams	5.3	95.3	99.4	100.0	100.0	100.0
nfl-teams	0.0	92.8	93.9	99.0	99.0	99.1
car-makers	71.6	53.6	76.1	79.6	84.8	86.5
us-presidents	0.0	36.4	34.6	59.5	59.5	59.5
us-states	74.9	96.9	98.0	99.9	99.9	100.0
japan-emperors	0.0	95.9	99.1	99.2	99.2	99.2
japan-prime-mins	13.1	71.3	91.7	93.1	93.0	93.0
japan-provinces	100.0	99.4	100.0	100.0	100.0	100.0
Average	**27.9**	**82.4**	**86.6**	**92.5**	**93.0**	**93.2**
Overall Average	**35.9**	**82.4**	**87.6**	**93.1**	**94.0**	**94.2**

Table 2.9: Performance (MAP) of Google Sets and various configurations of SEAL using three seeds.

English	**G.Sets**	**PE+RW**	**H1+RW**	**H2+RW**	**H3+RW**	**H4+RW**
disney-movies	25.0	78.2	75.7	76.5	53.1	52.3
constellations	28.9	100.0	100.0	100.0	99.9	99.9
countries	20.2	95.9	96.5	96.1	32.1	27.0
mlb-teams	94.0	100.0	100.0	100.0	91.1	82.1
nba-teams	100.0	100.0	100.0	100.0	76.7	66.9
nfl-teams	100.0	100.0	100.0	99.5	79.8	66.7
car-makers	57.0	93.0	93.2	93.4	46.1	33.0
us-presidents	92.1	100.0	100.0	100.0	99.7	99.7
us-states	80.8	100.0	100.0	100.0	99.2	98.6
cmu-buildings	7.9	48.8	30.9	30.9	28.6	28.6
diseases	6.6	20.9	20.0	19.5	15.1	14.5
periodic-comets	0.5	76.9	52.9	48.3	35.8	31.6
Average	51.1	84.5	80.8	80.4	63.1	58.4
Chinese	**G.Sets**	**PE+RW**	**H1+RW**	**H2+RW**	**H3+RW**	**H4+RW**
disney-movies	1.6	83.6	87.4	82.9	61.8	56.4
constellations	1.2	100.0	100.0	100.0	100.0	100.0
countries	9.3	96.5	96.6	96.6	93.2	93.7
mlb-teams	92.9	100.0	100.0	100.0	99.9	100.0
nba-teams	2.7	100.0	100.0	100.0	100.0	100.0
nfl-teams	0.6	99.7	99.2	99.1	84.5	84.5
car-makers	2.1	82.9	59.0	54.6	32.7	32.5
us-presidents	4.2	94.4	62.8	59.8	57.8	55.5
us-states	12.7	100.0	100.0	100.0	98.9	98.8
china-dynasties	4.9	62.9	64.3	63.3	40.6	40.6
china-provinces	81.0	98.9	99.0	99.3	94.5	93.3
taiwan-cities	99.3	100.0	100.0	100.0	99.2	98.7
Average	26.0	93.2	89.0	88.0	80.2	79.5
Japanese	**G.Sets**	**PE+RW**	**H1+RW**	**H2+RW**	**H3+RW**	**H4+RW**
disney-movies	14.4	77.2	75.5	70.0	79.8	75.9
constellations	1.6	100.0	100.0	100.0	100.0	100.0
countries	20.7	98.8	98.8	98.8	97.3	96.8
mlb-teams	2.7	99.6	99.7	99.8	99.8	99.7
nba-teams	20.0	100.0	100.0	100.0	99.8	99.3
nfl-teams	0.0	99.9	100.0	99.9	96.1	92.8
car-makers	41.0	90.3	89.5	88.4	85.3	83.1
us-presidents	1.0	97.1	95.2	94.1	96.5	96.6
us-states	58.4	100.0	100.0	100.0	99.8	99.8
japan-emperors	0.6	99.4	99.4	96.0	97.4	85.7
japan-prime-mins	4.8	95.0	99.3	99.3	85.8	82.1
japan-provinces	100.0	100.0	100.0	100.0	98.1	97.2
Average	22.1	96.4	96.5	95.5	94.6	92.4
Overall Average	**33.1**	**91.4**	**88.8**	**87.9**	**79.3**	**76.8**

Table 2.10: Performance (MAP) of Google Sets, our proposed extractor (PE), and the four types of HTML-based wrappers (H1-H4) using 200 web pages and only two seeds.

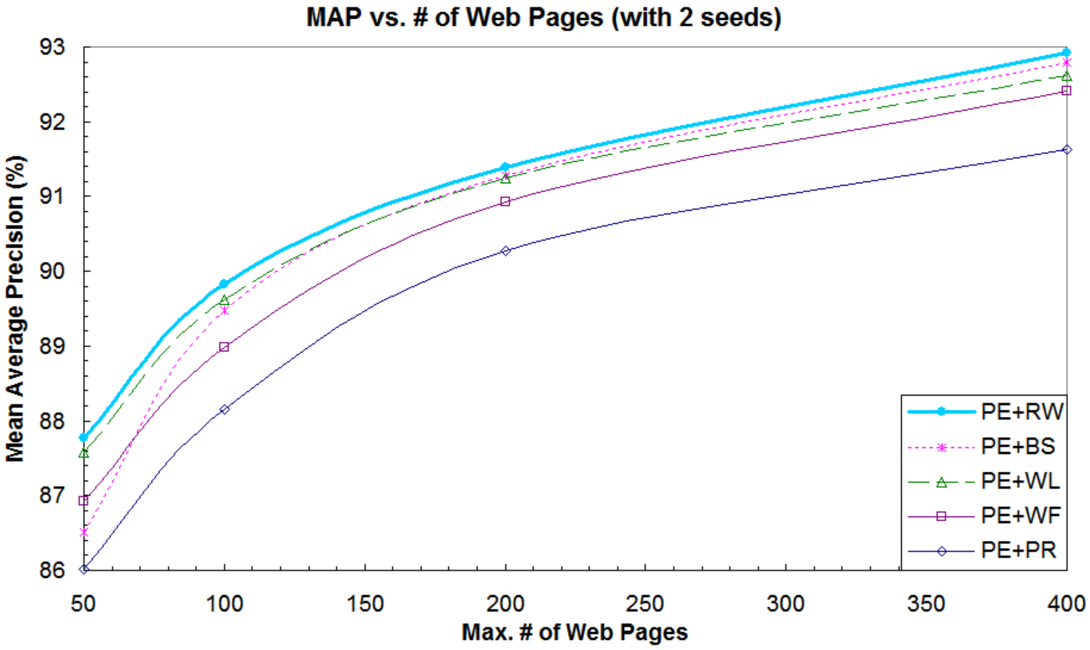


Figure 2.5: Performance (MAP) of alternative rankers on various number of web pages using two seeds.

2.6 Comparison to Prior Work

We present side-by-side qualitative comparison of set expansion results obtained by SEAL and published by two other research work: Talukdar *et al.* [41] and Ghahramani *et al.* [16].

2.6.1 Talukdar *et al.*, 2006

Talukdar *et al.* [41] present a context pattern induction method for named entity extraction. They extracted a fixed number N (context window size) of tokens immediately preceding and following the seed instances in their unlabeled data. From these extracted contexts, their system automatically selects *trigger words* to mark the beginning of a pattern, which is then used for bootstrapping from free text. By using this method, they extended several classes of seed entity lists into larger lists. They evaluated their approach on a newswire corpus which contains 31 million documents and showed improvement in accuracy of a conditional random field-based named-entity tagger.

Talukdar *et al.*		**SEAL**	
Rolex	Piaget	Omega	Wittnauer
Fossil	Raymond Weil	Cartier	Nike
Swatch	Girard Perregaux	Seiko	Oris
Cartier	Omega	Tag Heuer	Chopard
Tag Heuer	Guess	Ebel	IWC
Super Bowl	Frank Mueller	Rado	DKNY
Swiss	Citizen	Gucci	Wenger
Chanel	Croton	Bulova	Piaget
SPOT	David Yurman	Raymond Weil	Timex
Movado	Armani	Movado	ESQ
Tiffany	Audemars Piguet	Citizen	Guess
Sekonda	Chopard	Breitling	Patek Philippe
Seiko	**DVD**	Tissot	Croton
TechnoMarine	**DVDs**	Pulsar	Tommy Hilfiger
Rolexes	**Chinese**	Fossil	Sector
Gucci	Breitling	Hamilton	Invicta
Franck Muller	Montres Rolex	Rolex	Oakley
Harry Winston	Armitron	Casio	Skagen
Patek Philippe	Tourneau	Swatch	Anne Klein
Versace	**CD**	Concord	Armitron
Hampton Spirit	**NFL**	Swiss Army	Zodiac

Table 2.11: Set Comparison: Watch Brands

Table 2.11 illustrates the top 42 set expansion results by Talukdar *et al.* on watch brand names using 17 seeds. As comparison, we present our top 42 results in the right column using only the first three of their 17 seeds (i.e., *Corum*, *Longines*, and *Lorus*). By examining the top 42 extracted instances, their system returned noisy instances (underlined items) towards the bottom of their list and achieved a precision of 85.7%; whereas, SEAL achieved a precision of 100% by using only the first three of their 17 seeds. We also tried the three seeds on Google Sets but obtained no results other than the seeds themselves.

2.6.2 Ghahramani *et al.*, 2005

Ghahramani *et al.* [16] evaluated their Bayesian Sets algorithm on three datasets, including the EachMovie dataset, consisting of movie ratings by users of the EachMovie service, and the NIPS authors dataset, consisting of the text of articles published in

Seeds: Mary Poppins, Toy Story

Google Sets	Bayesian Sets	SEAL
Toy Story	Mary Poppins	Mary Poppins
Mary Poppins	Toy Story	Toy Story
Mulan	Winnie the Pooh	Cinderella
Toy Story 2	Cinderella	Hercules
Moulin Rouge	The Love Bug	The Lion King
Monsters Inc	Bedknobs and Broomsticks	Pocahontas
Man on the Moon	Davy Crockett	Pinocchio
Mummy The	The Parent Trap	Beauty and the Beast
Matrix The	Dumbo	The Jungle Book
Mod Squad The	The Sound of Music	Song of the South

Table 2.12: Set Comparison: Children's Movies

Seeds: L. Saul, T. Jaakkola

Bayesian Sets		SEAL	
L. Saul	T. Jebara	T. Jaakkola	C. Bishop
T. Jaakkola	W. Wiegerinck	L. Saul	M. I. Jordan
M. Rahim	M. Meila	B. Frey	Z. Ghahramani
M. Jordan	S. Ikeda	P. Niyogi	A. Smola
N. Lawrence	D. Haussler	M. J. Wainwright	Y. Weiss

Table 2.13: Set Comparison: NIPS Authors

NIPS volumes 0-12. They pre-processed the EachMovie dataset by removing movies rated by less than 15 people, and people who rated less than 200 movies. The dataset was binarized so that a (person, movie) entry had a value of 1 if the person gave the movie a rating above 3 stars (from a possible 0-5 stars). The NIPS author dataset was pre-processed and binarized by column normalizing each author, and then thresholding so that a (word, author) entry is 1 if the author uses that word more frequently than twice the word mean across all authors.

Table 2.12 shows top 10 set expansion results on children's movies from Bayesian Sets, as presented in Ghahramani *et al.* We provide results from Google Sets and SEAL as comparisons. Note that Bayesian Sets used a movie-specific dataset: EachMovie. As illustrated, both Bayesian Sets and SEAL systems perform well on finding children movies. Similar to Table 2.12, Table 2.13 shows top 10 results on NIPS authors from Bayesian Sets as presented in Ghahramani *et al.*. We provide results from SEAL as comparisons but not from Google Sets since it failed to return any result. Note that Bayesian Sets, again, used a domain-specific dataset: the NIPS dataset. Although the algorithm was shown to rank instances effectively, it solves only a particular subproblem of set expansion (i.e. the ranking problem), in which candidate instances (e.g., actor, movie, author names) are provided.

2.7 Summary

In this chapter, we have proposed a novel graph-based approach to set expansion using semi-structured documents at the character-level. Based on our proposed approach, we developed an open-source set-expansion system called SEAL that is independent of both human and mark-up language. We have shown that SEAL is capable of handling various languages such as English, Chinese, and Japanese. By conducting ablation studies using SEAL, we have also shown that our random walk approach outperforms four other alternative rankers, including Bayesian Sets and PageRank, for the problem of set expansion. The studies also show that our novel character-based wrapper induction technique is more effective than a simpler technique that is often used by other researchers. Our experimental results illustrate that character-based wrappers are better suited than HTML-based wrappers for the task of set expansion. This is supported by the following two experimental results: 1) SEAL outperforms Google Sets, a web-based set expansion system that exploits HTML structure, in terms of mean average precision for the 36 datasets tested, and 2) the more we restrict wrappers to HTML-based, the worse the set expansion performance.

Chapter 3

Noise Resistant SEAL (for List Question Answering)

3.1 Introduction

Question answering (QA) systems are designed to retrieve precise answers to questions posed in natural language. A *list question*[1] expects a list as its answer, e.g. *Name the coffee-producing countries in South America.* The ability to answer list questions has been tested as part of the yearly TREC QA evaluation [12, 13]. This chapter focuses on the use of *set expansion* to improve list question answering. We explore the hypothesis that a set expansion algorithm, when carefully designed to handle noisy inputs, can be applied to the output from a QA system to produce an overall list of answers for a given question that is better than the answers produced by the QA system itself.

We propose to exploit large, redundant sources of structured and/or semi-structured data and use linguistic analysis to seed a shallow analysis of these sources. This is a hard problem since the linguistic evidence used as seeds is noisy. More precisely, we combine the QA system Ephyra [36] with the set expansion system SEAL to create a hybrid approach that performs better than either system by itself when tested on data from the TREC 13-15 evaluations. In addition, we apply our set expansion algorithm to answers generated by the five QA systems that performed the best on the list questions in the TREC 15 evaluation and report improvements in F_1 scores for four of these systems [50].

[1] See appendix C for all list-type questions used in our experiments.

In this chapter, Section 3.2 gives an overview of the QA system used for our experiments. Section 3.3 describes how SEAL was adapted to deal with noisy seeds produced by QA systems, and Section 3.4 presents the design and results of the experiments. Lastly, we summarize this chapter in Section 3.5.

3.2 Ephyra Question Answering System

We chose the QA system Ephyra [35, 36] for our experiments because it scored well in the TREC QA track [12, 13] and we have access to all its internal resources, including confidence scores for each candidate answer it produces, which are crucial elements for our proposed approach. The system combines three answer extraction techniques for list questions: (1) an answer type classification approach; (2) a syntactic pattern learning and matching approach; and (3) a semantic extractor that uses a semantic role labeling system. The answer type based extractor classifies questions by their answer types and extracts candidates of the expected types. The Ephyra pattern matching approach learns textual patterns that relate question key terms to possible answers and applies these patterns to candidate sentences to extract factoid answers. The semantic approach generates a semantic representation of the question that is based on predicate-argument structures and extracts answer candidates from similar structures in the corpus. The source code of the answer extractors is included in OpenEphyra, an open source release of the system.[1]

The answer candidates from these extractors are combined and ranked by a statistical answer selection framework [18], which estimates the probability of an answer based on a number of answer validation and similarity features. Validation features use resources such as gazetteers and Wikipedia to verify an answer, whereas similarity features measure the syntactic and semantic similarity to other candidates, e.g. using string distance measures and WordNet relations.

3.3 Proposed Approach

We want to apply set expansion to answer candidates for list questions generated by Ephyra and other TREC QA systems to find additional instances of correct answers

[1] `http://www.ephyra.info/`

that were not in the original candidate set. However, SEAL was originally designed to handle only relevant input seeds. When provided with a mixture of relevant and irrelevant answers from a QA system, performance suffers. In this section, we propose three modifications to SEAL to improve its ability to handle noisy input seeds.

3.3.1 Aggressive Fetcher

For each expansion, SEAL's fetcher concatenates all seeds and sends them as one query to the search engines (i.e., Google and Yahoo!). However, when the seeds are noisy, the documents fetched are constrained by the irrelevant seeds, which decreases the chance of finding good documents (i.e., documents containing correct answers). To overcome this problem, we designed an *aggressive fetcher* (AF) that increases the chance of composing queries containing only relevant seeds. It sends a two-seed query for every possible pair of seeds to the search engines. If there are n input seeds, then the total number of queries sent would be $\binom{n}{2}$.

For example, suppose SEAL is given a set of noisy seeds: *Boston*, *Seattle* and *Carnegie-Mellon* (assuming *Carnegie-Mellon* is irrelevant), then by using AF, the queries will be "Boston Seattle", "Boston Carnegie-Mellon", and "Seattle Carnegie-Mellon", where one query (i.e., "Boston Seattle") will contain only relevant seeds (as shown in Table 3.1). The documents are then collected and sent to SEAL's extractor for learning wrappers.

	Queries	**Quality**
-AF	#1: Boston Seattle Carnegie-Mellon	Low
+AF	#1: Boston Seattle	High
	#2: Boston Carnegie-Mellon	Low
	#3: Seattle Carnegie-Mellon	Low

Table 3.1: Example queries and their quality given the seeds *Boston*, *Seattle* and *Carnegie-Mellon*, where *Carnegie-Mellon* is assumed to be irrelevant.

3.3.2 Lenient Extractor

SEAL's extractor requires the longest common contexts to bracket at least one instance of every seed per web page. However, when seeds are noisy, such common contexts usually do not exist or are too short to be useful. To solve this problem, we propose a *lenient extractor* (LE) which only requires the contexts to bracket at least one instance of *a minimum of two* seeds, instead of *every* seed. This increases the chance of finding longest common contexts that bracket only relevant seeds. For instance, suppose SEAL is given the seeds from the previous example (*Boston*, *Seattle* and *Carnegie-Mellon*) and the passage below. Then the extractor would learn the wrappers shown in Table 3.2.

"While attending a hearing in Boston City Hall, Alan, a professor at Boston University, met Tina, his former student at Seattle University, who is studying at Carnegie-Mellon University Art School and will be working in Seattle City Hall."

	Learned Wrappers
-LE	#1: `at [...]  University`
+LE	#1: `at [...]  University` #2: `in [...]  City Hall`

Table 3.2: Wrappers learned by SEAL's extractor when given the passage above and the seeds: *Boston*, *Seattle*, and *Carnegie-Mellon*.

As illustrated, with lenient extraction, SEAL is now able to learn the second wrapper because it brackets one instance of at least two seeds (*Boston* and *Seattle*). This can be very helpful if the list question is asking for city names rather than university names. The extractor then uses these wrappers to extract additional answer candidates, by searching for other strings that fit into the placeholders of the wrappers. Note that the example was simplified for ease of presentation. The wrappers are actually character-based (as opposed to word-based) and are likely to contain HTML tags when generated from real web pages.

3.3.3 Hinted Expander

Most QA systems use keywords from the question to guide the retrieval of relevant documents and the extraction of answer candidates. We believe these keywords are also important for SEAL to identify additional instances of correct answers. For example, if a question asks for all the U.S. presidents, and the seeds are *George Washington*, *John Adams*, and *Thomas Jefferson*, then without using any context from the question, SEAL would output a mixture of founding fathers and presidents of the U.S.A. To solve this problem, we devised a *hinted expansion* (HE) technique that utilizes the context given in the question to constrain SEAL's search space on the Web. This is achieved by appending keywords (e.g., presidents) from the question to every query that is sent to the search engines. The rationale is that the retrieved documents will also match the keywords, which may increase the chance of finding those documents that contain our desired set of answers. Preliminary experiments showed that we can obtain a good balance between the amount and quality of the documents fetched by using three question keywords as hint words. In particular, we select and use as hints the question keywords that occur least frequently in a sample of the AQUAINT newswire corpus used in the TREC evaluations.

Below, we show the question keywords, Ephyra's answers (first four were used as seeds to SEAL), AF+HE's queries, and SEAL's results for TREC question #143.6. For this question, the average precision of Ephyra's answers is 35.2% and SEAL's answers is 48.3%.

Question: Who have been "scholars" at the Institute (American Enterprise Institute)?
Question Keywords: Institute, scholars, American Enterprise Institute
Ephyra's Answers: michael ledeen, sidney blumenthal, ronald reagan, norman ornstein, norm ornstein, christina hoff sommers, rosa delauro, christina hawke smith, michael novak, robert bork, . . .

AF+HE's Queries:

1. "michael ledeen", "sidney blumenthal", Institute scholars American Enterprise Institute
2. "michael ledeen", "ronald reagan", Institute scholars American Enterprise Institute
3. "michael ledeen", "norman ornstein", Institute scholars American Enterprise Institute
4. "sidney blumenthal", "ronald reagan", Institute scholars American Enterprise Institute
5. "sidney blumenthal", "norman ornstein", Institute scholars American Enterprise Institute
6. "ronald reagan", "norman ornstein", Institute scholars American Enterprise Institute

SEAL's Answers: michael novak, michael ledeen, ronald reagan, norman ornstein, christina hoff sommers, joshua muravchik, bill clinton, robert bork, sally satel, richard perle, . . .

3.4 Experiments

We conducted experiments in two phases. In the first phase, we evaluated the set expansion approach by applying SEAL to answers generated by Ephyra. In the second phase, we evaluated the approach by applying SEAL to the output from QA systems that performed the best on the list questions in the TREC 15 evaluation. In both phases, the answers found by SEAL were retrieved from the Web instead of the AQUAINT corpus. However, we rejected answers if they could only be found in the Web and not in the AQUAINT corpus, to avoid giving SEAL an unfair advantage over the QA systems: TREC participants were allowed to extract candidates from the Web (or any other source), but they had to identify a supporting document in the AQUAINT corpus for each answer, and thus could not return answers that were not covered by the corpus.

The candidate answers were evaluated by using the answer keys, composed of regular expression patterns, obtained from the TREC website. These answer keys were served as a reference and were not officially used in the TREC evaluation[1], thus the baseline scores we computed for Ephyra and other QA systems in our experiments are slightly different from those officially reported. Due to time constraint, we did not extend the patterns with additional correct answers found in our experiments.

3.4.1 Experiment with Ephyra

We evaluated our set expansion approach on Ephyra using the list questions from TREC 13, 14, and 15. The table below shows the number of questions present in each TREC. For each question, the top four answer candidates from Ephyra were given as input seeds to SEAL. Initial experiments showed that by adding additional seeds, the effectiveness of our approach can be improved at the expense of a longer runtime.

TREC	# of Questions	ID Range
13	55	1∼65
14	93	66∼140
15	89	141∼215

We report mean average precision (MAP) as well as F_1 score, which is an evaluation metric commonly used in the QA field. For the F_1 scores, we drop answer candidates with low confidence scores by applying a cut-off threshold: an answer candidate is dropped if its confidence score is below a threshold. Here, we define an *optimal threshold* for a question as the threshold that maximizes the F_1 score for that particular question.

[1] The TREC evaluated candidate answers based on several factors, including human judgments and an undisclosed set of correct answers.

TREC	**Ephyra**	**Ephyra's Top 4 Ans**	**SEAL**	**SEAL +LE**	**SEAL +LE+AF**	**SEAL +LE+AF+HE**	
13	26.0	21.4	23.8	31.4	34.2	35.3	Δ35.9%
14	14.5	8.7	14.5	17.0	16.6	18.8	Δ30.2%
15	13.4	9.0	13.2	16.9	17.1	18.9	Δ41.2%
Avg	17.9	13.0	17.1	21.8	22.6	24.3	Δ35.7%

Table 3.3: Mean average precision of Ephyra, its top four answers, and various SEAL configurations, where LE is Lenient Extractor, AF is Aggressive Fetcher, and HE is Hinted Expander. The last column shows the best-configured SEAL's relative improvements over Ephyra.

TREC	**Ephyra**	**Ephyra's Top 4 Ans**	**SEAL**	**SEAL +LE**	**SEAL +LE+AF**	**SEAL +LE+AF+HE**	
13	35.7	26.3	30.5	36.5	40.1	40.8	Δ14.1%
14	22.8	14.1	20.6	22.8	22.7	24.9	Δ9.0%
15	22.4	14.6	19.9	23.3	24.0	25.7	Δ14.4%
Avg	27.0	18.3	23.7	27.5	28.9	30.4	Δ12.8%

Table 3.4: Average F_1 of Ephyra, its top four answers, and various SEAL configurations when using an optimal threshold for each question. The last column shows the best-configured SEAL's relative improvements over Ephyra.

For each TREC dataset, we conducted three experiments: (1) evaluation of answer candidates using MAP; (2) evaluation using average F_1 with an optimal threshold for each question; and (3) evaluation using average F_1 with thresholds trained by 5-fold cross validation. For each of those 5-fold validations, only one threshold was determined for all questions in the training folds.

In Tables 3.3 and 3.4, we present evaluation results for all answers from Ephyra, only the top four answers, and various configurations of SEAL using the top four answers as seeds. Table 3.3 shows the MAP for each dataset (TREC 13, 14, and 15), and Table 3.4 shows for each dataset the average F_1 score when using optimal per-question thresholds. The results indicate that SEAL achieves the best performance when configured with all three proposed extensions. In terms of MAP, the best-configured SEAL improves the quality of the input answers (relatively) by 65%, 116%, 110% for each dataset respectively. In terms of optimal F_1, SEAL improves the quality of the input answers by 55%, 77%, 76% respectively. These results illustrate that SEAL performs the best on noisy inputs when it is configured to incorporate all proposed modifications: AF,

TREC	**Ephyra**		**Best SEAL**		**Hybrid**		
	Avg F_1	**Avg Thr**	**Avg F_1**	**Avg Thr**	**Avg F_1**	**Avg Thr**	ΔF_1
13	25.5	0.381	30.7	0.326	29.0	0.080	13.7%
14	15.8	0.264	15.6	0.189	17.1	0.011	8.5%
15	15.2	0.119	15.6	0.258	16.5	0.012	8.4%
All	18.0	0.288	19.2	0.261	19.6	0.016	8.6%

Table 3.5: Average F_1 of Ephyra, the best-configured SEAL (SEAL+LE+AF+HE), and the hybrid system, along with thresholds trained by 5-fold cross validation. The last column shows Hybrid's relative improvements over Ephyra.

LE, and HE. They also illustrate that a set expansion system is capable of improving a QA system's performance on list questions, if we know how to select good thresholds.

In practice, the thresholds are unknown and must be estimated from a training set. Table 3.5 shows evaluation results using 5-fold cross validation for each dataset (TREC 13, 14, and 15) independently, and the combination of all three datasets (All). For each validation, we determine the threshold that maximizes the F_1 score on the training folds, and we also determine the F_1 score on the test fold by applying the trained threshold. We repeat this validation for each of the five test folds and present the average threshold and F_1 score for each configuration and dataset. The F_1 scores give an estimate of the performance on unseen data and allow a fair comparison across systems. Here, we also introduce a hybrid system (Hybrid) that intersects the answers found by both systems by multiplying their probabilistic scores.

Tables 3.3, 3.4, and 3.5 show that the effectiveness of the set expansion approach depends on the quality of the initial answer candidates. The improvements are most apparent for the TREC 13 dataset, where Ephyra has a much higher performance compared to TREC 14 and 15. However, the best-configured SEAL did not improve the F_1 score on TREC 14, as reported in Table 3.5. We suspect that this is due to the comparatively low quality of Ephyra's top four answers for this dataset. The experiments also illustrate that by intersecting the answer candidates found by Ephyra and SEAL, we can eliminate poor answer candidates and partially compensate for the low precision of Ephyra on the harder TREC datasets. However, this comes at the expense of a lower recall, which slightly hurts the performance on the comparatively easier TREC 13 questions. We also evaluated Google Sets on top four answers from

TREC 15 Systems	Baseline Avg F_1	Top 4 Avg F_1	Google Sets Avg F_1	 ΔF_1	Best SEAL Avg F_1	 ΔF_1	Hybrid Avg F_1	 ΔF_1
lccPA06	45.0	32.7	37.9	-15.7%	40.0	-11.0%	45.3	0.8%
cuhkqaepisto	18.3	17.0	16.0	-12.7%	19.7	8.1%	19.1	4.7%
NUSCHUAQA1	18.4	15.0	16.7	-9.2%	18.7	1.9%	18.1	-1.8%
FDUQAT15A	19.7	14.3	18.8	-4.6%	19.8	0.4%	20.6	4.6%
QACTIS06C	17.5	15.2	17.0	-2.7%	18.4	5.3%	18.4	4.9%
Average	23.8	18.8	21.3	-10.5%	23.3	-1.8%	24.3	2.2%

Table 3.6: Average F_1 of the top QA systems, their top four answers, Google Sets, the best-configured SEAL (SEAL+LE+AF+HE), the hybrid system, and their relative improvements over the QA systems.

Ephyra for TREC 13-15 and obtained F_1 scores of 12%, 11%, and 9% respectively (compared to 29%, 17%, and 16% for our hybrid approach with trained thresholds).

3.4.2 Experiment with Top QA Systems

We evaluated two set expansion approaches, SEAL and Google Sets, on the five QA systems that performed the best on the list questions in TREC 15. For each question, the top four answer candidates[1] from those systems were given as input seeds to SEAL and Google Sets. Unlike the candidates found by Ephyra, these candidates were provided without confidence scores; hence, we assumed they all have a score of 1.0. In our experiments with SEAL, we first determined a single threshold that optimizes the average of the F_1 scores of the top five systems in both TREC 13 and 14. We then obtained evaluation results for the top systems in TREC 15 by using this trained threshold. When performing hinted expansion, the keywords (or hint words) for each question were extracted by Ephyra's question analysis component. In our experiments with Google Sets, we requested *Small Sets* of items and again measured the performance in terms of F_1 scores. We also tried requesting *Large Sets* but the results were worse.

Table 3.6 shows F_1 scores for the set expansion approach applied to the output from the five QA systems with the highest performance on the list questions in TREC 15. Again, Hybrid intersects the answers found by the QA system and SEAL by multiplying their confidence scores. Two thresholds were trained separately on the top five systems

[1] Obtained from `http://trec.nist.gov/results`

Question 154.6: Name titles of movies, other than "Superman" movies, that Christopher Reeve acted in.

lccPA06 (F_1: 75%)	SEAL+LE+AF+HE (F_1: 40%)
+Rear Window	+Rear Window
+The Remains of the Day	+The Remains of the Day
+Snakes and Ladders	−The Bostonians
−Superman	−Somewhere in Time
	−Village of the Damned
	−In the Gloaming

Table 3.7: Example of SEAL being penalized for finding correct answers (all are correct except the last one). Answers found in the answer keys are marked with "+". All four answers from "lccPA06" were used as seeds.

Question 170.6: What are the titles of songs written by John Prine?

NUSCHUAQA1 (F_1: 25%)	SEAL+LE+AF+HE (F_1: 44%)
+I Just Want to Dance With You	+I Just Want to Dance With You
−Titled In Spite of Ourselves	+Christmas in Prison
+Christmas in Prison	+Sam Stone
−Grammy - Winning	−Grandpa was a Carpenter
	−Sabu Visits the Twin Cities Alone
	+Angel from Montgomery

Table 3.8: Example demonstrating SEAL's ability to handle noisy input seeds. All four answers from "NUSCHUAQA1" were used as seeds. Again, SEAL is penalized for finding correct answers (all answers are correct).

in both TREC 13 and 14; one for SEAL (0.2376) and another for Hybrid (0.2463). As shown, the performance of Google Sets is worse than SEAL and Hybrid, but better than the top four answers on average.

The results show that both SEAL and Hybrid are capable of improving four out of the five systems. We observed that one reason why SEAL did not improve "lccPA06" was the incompleteness of the answer keys. Table 3.7 shows one of many examples where SEAL was penalized for finding additional correct answers. As illustrated, Hybrid improved all systems except "NUSCHUAQA1". The reason is that even though SEAL improved the baseline, their overlapping answer set is too small, thus hurting the recall of Hybrid substantially. Unfortunately, for the top TREC 15 systems we only had access to the answers that were actually submitted by the participants, whereas for Ephyra we could utilize the entire list of generated answer candidates, including those that fell below the cutoff threshold for list questions. Nevertheless, the hybrid approach could improve the baseline by more than 2% on average in terms of F_1 score. Table 3.8 shows that the best-configured SEAL is capable of expanding only the relevant seeds even when given a set of noisy seeds. Neither Google Sets nor the original set expansion algorithm without the proposed extensions could expand these seeds with additional candidates.

3.5 Summary

In this chapter, we have shown that SEAL is capable of improving the performance of question answering (QA) systems on list questions by utilizing only their top four answer candidates as seeds. We have also illustrated a feasible and effective method for integrating a set expansion approach into any QA system. We would like to emphasize that for each of the experiments we conducted in this chapter, all that SEAL received as input were the top four noisy answers from a QA system and three keywords from the TREC questions. We have shown that higher quality candidates support more effective set expansion. In the future, we will investigate how to utilize more answer candidates from the QA system and determine the minimal quality of those candidates required for SEAL to make an improvement.

We have also shown that, in terms of F_1 scores with trained thresholds, the hybrid method improves the Ephyra QA system on all datasets and also improves four out of the five systems that performed the best on the list questions in TREC 15. However, the final list of answers only comprises candidates found by both the QA system and SEAL. In future experiments, we will investigate other methods of merging answer candidates, such as taking the union of answers from both systems. We expect further improvements from adding candidates that are found only by the QA system, but it is unclear how the confidence measures from the two systems can be combined effectively. We would also like to emphasize that SEAL is entirely language-independent, and thus can be readily applied to answer candidates in other languages. In future experiments, we will investigate its performance on question answering tasks in languages such as Chinese and Japanese.

As pointed out previously, the performance of SEAL highly depends on the accuracy of the seeds. However, QA systems are usually not optimized to provide few high-precision results, but treat precision and recall as equally important. This leaves room for further improvements, such as applying stricter answer validation techniques to the seeds used for set expansion. We also plan to analyze the effectiveness of our approach across different question types and evaluate it on more complex questions such as the rigid list questions in the new TAC QA evaluation, which ask for opinion holders and subjects.

Chapter 4

Iterative SEAL

4.1 Introduction

Although SEAL works well given two or three seeds, it has a limitation on the number of seeds it can handle. Table 4.1 shows the performance of SEAL (as the MAP score averaged across 36 datasets) for four different rankers when provided with two to six *supervised seeds* (i.e., correct seeds) randomly selected from our development set. As shown, SEAL performs the best when given four seeds, but its performance drops substantially when given more than five seeds. One reason is that the more query words submitted to a search engine, the fewer the returned web pages. Since SEAL's first step is to retrieve web pages containing all seeds, the more seeds provided to SEAL, the fewer the web pages it can analyze and utilize. To overcome this limitation, we introduce a system called Iterative SEAL (iSEAL), which uses an (supervised) iterative process that performs *supervised expansion* multiple times. In each iteration, iSEAL invokes SEAL on a few supervised seeds, and statistics are accumulated from iteration to iteration to obtain a final ranking.

Bootstrapping is an (unsupervised) iterative process in which a system continuously consumes its own outputs to improve its own performance [15, 19, 25]. The advantage of bootstrapping is that, if successful, the performance of a system can be greatly improved with minimal supervision, so it is natural to explore bootstrapping using iSEAL. We propose a bootstrapping technique that requires only two supervised seeds, which are used to trigger the first expansion of the iterative process above. In each iteration after the first, iSEAL expands a few *unsupervised seeds* (i.e., highly ranked

	# Seeds (seed size)				
Ranker	**2**	**3**	**4**	**5**	**6**
Random Walk (RW)	77.1	83.9	**84.5**	83.7	78.9
PageRank (PR)	74.1	82.6	**83.4**	83.0	78.5
Bayesian Sets (BS)	77.0	84.1	**84.8**	84.0	79.3
Wrapper Length (WL)	77.5	83.2	**83.3**	82.2	78.0
Average	76.4	83.5	**84.0**	83.2	78.7

Table 4.1: MAP of set expansion using various rankers and various number of seeds on our development set. Note that four seeds maximize the set expansion performance.

items obtained in the previous iteration of iSEAL), and again statistics are accumulated from iteration to iteration.

Bootstrapping introduces a potential problem, as the self-provided seeds used in bootstrapping may be incorrect, and prior results do not indicate how SEAL performs with "noisy" seeds. We show that iSEAL, when used in bootstrap mode, is indeed much more sensitive to the choice of rankers and number of seeds. We compare several rankers, including Random Walk with Restart [43]; PageRank [30], which was designed to rank hyperlinked documents; Bayesian Sets [16], which formulates the set expansion problem as a Bayesian inference problem; and a fast but ad hoc ranking heuristic we call Wrapper Length. Please refer to Section 2.4.3 for the details of Random Walk and Section 2.5.2 for the three other rankers.

In this chapter, Section 4.2 presents Iterative SEAL. Section 4.3 describes the details of the experimental design, and Section 4.4 presents the experimental results. We summarize this chapter in Section 4.5.

4.2 Iterative SEAL

In this section, we present the Iterative SEAL (iSEAL) system and consider two different iterative processes: supervised expansion and bootstrapping. In our experiments, both processes start their first iteration with two supervised seeds, which is the smallest number of seeds required by the wrapper induction technique used in SEAL. In every successive iteration, iSEAL expands a couple of seeds while accumulating statistics from one iteration to another; thus no information is ever tossed away. This allows the expansion of seeds in the current iteration to have access to all statistics computed in the past iterations. Below, we summarize the approach each ranker utilizes for accumulating statistics. Note that the ranker, Wrapper-Frequency (WF), simply ranks each candidate instance i by the number of wrappers that extract i.

Ranker	**Approach for Accumulating Statistics**
RW & PR	Attach additional nodes and edges to the existing graph
BS	Append new rows and columns to the existing feature table
WL	Add new score to the existing score of an instance
WF	Add new frequency count to the existing count of an instance

In every iteration, there are several ways to select seeds. For each process, we propose two seeding (i.e., seed selection) strategies: Fixed Seed Size (FSS) and Increasing Seed Size (ISS). In Section 4.4, we evaluate each ranker using the two iterative processes (i.e., supervised and bootstrapping) and the two seeding strategies (i.e., FSS and ISS). We show that the seeding strategy has significant influence on the overall performance of the system, and that the ranker is more important when seeds are noisy.

4.2.1 Iterative Supervised Expansion

The iterative supervised expansion improves SEAL's performance by allowing it to handle an unlimited number of supervised seeds. In each iteration, iSEAL expands a couple of randomly selected supervised seeds while accumulating statistics from one iteration to another. In this section, we present the two seeding strategies for this process.

4.2.1.1 Fixed Seed Size

Fixed Seed Size (FSS) is a strategy that requires two seeds in every iteration. In supervised expansion, the two seeds are randomly selected from a collection of seeds provided by some reliable source (i.e., users). The pseudo-code for this strategy is presented below:

```
$stats \leftarrow \varnothing$
for $i = 1$ to M do
    $seeds \leftarrow select_2(E)$
    $stats \leftarrow expand(seeds, stats)$
    $list \leftarrow rank_r(stats)$
end for
```

where M is the total number of iterations (inclusively), $select_n(E)$ randomly selects n different seeds from the set E, E is a set containing supervised seeds, $expand(seeds, stats)$ expands the selected *seeds* using *stats* and outputs accumulated statistics, and $rank_r(stats)$ applies the ranker r on the accumulated *stats* to produce a ranked *list* of candidate instances. In our experiments, we evaluate the quality of the ranked *list* at each M.

4.2.1.2 Increasing Seed Size

Increasing Seed Size (ISS) is a strategy that starts the iterative process with two supervised seeds, then increments the number of seeds by one for every successive expansion, until a maximum size of n is reached. After then, it continues to expand using n seeds. We set n to be four based on results in Table 4.1, which shows that four seeds maximize the set expansion performance. This number has also been used by Etzioni et al. [15] and Nadeau et al. [25]. The pseudo-code for this strategy is presented below:

```
$stats \leftarrow \varnothing, used \leftarrow select_1(E)$
for $i = 1$ to M do
    $m = min(3, |used|)$
    $seeds \leftarrow select_m(used) \cup select_1(E)$
    $stats \leftarrow expand(seeds, stats)$
    $list \leftarrow rank_r(stats)$
    $used \leftarrow used \cup seeds$
end for
```

where *used* is a set that contains previously expanded seeds and $min(x, y)$ returns the minimum of x and y. All other notations are the same as those defined in the last section. This strategy starts by expanding two supervised seeds. For the second iteration, it expands three seeds: two previously-used seeds plus an additional new supervised seed. For every successive iteration, it expands four seeds: three randomly selected previously-used seeds plus a new supervised seed. As a result, every iteration introduces a new supervised seed, and every iteration after the second one consumes a total of four seeds (three previously-used and one new seed).

4.2.2 Bootstrapping

Bootstrapping refers to iterative unsupervised set expansion. It is an iterative process for a system to continuously improve its own performance by utilizing its own outputs. Unlike supervised expansion, this process requires minimal supervision, but is very sensitive to the system's performance because errors can easily propagate from one iteration to another. For example, if the top ranked entities in the last iteration are incorrect, then when they are chosen as seeds, the expansion results of the next iteration could be worse. Carefully designed seeding strategies can minimize the propagated errors. We present the two seeding strategies in the unsupervised mode.

4.2.2.1 Fixed Seed Size

As mentioned earlier, FSS is a strategy that requires two seeds in every iteration. Unlike supervised expansion which expands supervised seeds, bootstrapping expands unsupervised seeds that are the most confident new instances (i.e., instances that has not been used as seeds) extracted from the last iteration. The pseudo-code for this strategy is presented below:

```
$stats \leftarrow \varnothing, seeds \leftarrow select_2(E)$
for $i = 1$ to M do
  if $i > 1$ then
    $seeds \leftarrow top_2(list)$
  end if
  $stats \leftarrow expand(seeds, stats)$
  $list \leftarrow rank_r(stats)$
end for
```

where $top_2(list)$ returns a new pair of instances which has the highest joint probabilistic weights according to their confidence scores in the ranked *list* of last iteration. More specifically, for every successive i^{th} iteration after the first expansion, this strategy selects, from the results of $(i-1)^{th}$ iteration, a new pair of instances to be used as seeds for the i^{th} iteration. Note that regardless of the number of iterations, the two initial seeds are the only supervised seeds required.

4.2.2.2 Increasing Seed Size

In the unsupervised mode of ISS, the strategy is exactly the same as in the supervised mode, except that after the first iteration, the new seed (i.e., the instance never used as seed) at every i^{th} iteration is the highest-ranked new instance in the $(i-1)^{th}$ iteration. The pseudo-code is presented below:

```
$stats \leftarrow \varnothing, used \leftarrow select_2(E)$
for $i = 1$ to M do
  $m = min(3, |used|)$
  $seeds \leftarrow select_m(used)$
  if $i > 1$ then
    $seeds \leftarrow seeds \cup top_1(list)$
  end if
  $stats \leftarrow expand(seeds, stats)$
  $list \leftarrow rank_r(stats)$
  $used \leftarrow used \cup seeds$
end for
```

where $top_1(list)$ returns a new instance that has the highest weight in the ranked *list* of last iteration. Again, the two initial seeds are the only supervised seeds required.

4.3 Experimental Setting

We evaluate the iterative processes using the datasets described in Section 2.5.4, the evaluation metric – Mean Average Precision described in Section 2.5.5, the four alternative rankers – Bayesian Sets, PageRank, Wrapper Length, and Wrapper Frequency described in Section 2.5.2, and our proposed ranker – Random Walk described in Section 2.4.3. Wrapper Frequency is used as the baseline in the experimental results.

Each experiment evaluates a particular combination of iterative process, seeding strategy, and ranker; we evaluated all 20 possible combinations. For each combination, we performed ten iterative expansions on each of the 36 evaluation datasets independently three times; thus, at each of the ten iterations, there are 108 (3×36) ranked lists. We then report the MAP of those ranked lists in the experimental results.

4.4 Experimental Results

We first examine the effect of supervised expansion. Figure 4.1 and 4.2 illustrate the performance of various rankers using supervised expansion with FSS and ISS respectively. The error bars in those figures are the standard errors of the mean, and to simplify the graph we only show error bars for Random Walk (standard errors for the other methods are comparable). Although both strategies improve the performance of the rankers, FSS improves faster than ISS. The reason is that FSS requires two new supervised seeds at every iteration whereas ISS requires only one. From the two graphs, we observe that Bayesian Sets (BS) performs the best, Random Walk (RW) is nearly as good as Bayesian Sets, Wrapper Frequency (WF) is almost as good as Wrapper Length (WL), and PageRank (PR) is the worst among the five rankers.

We then examine the effect of bootstrapping. Figure 4.3 and 4.4 illustrate the performance of various rankers in bootstrap mode using FSS and ISS respectively. As illustrated, ISS reliably improves with more seeds, but FSS failed to improve any ranker other than Random Walk at the 10^{th} iteration, and even this improvement was modest. This result shows that bootstrapping set expansion is not a trivial task. The ISS strategy was carefully designed to circumvent this performance problem. While ISS uses only two supervised seeds, it is much more conservative about using self-generated seeds: at every expansion, FSS boldly introduces two new seeds taken from the results of the last iteration, whereas ISS conservatively introduces only one. Therefore, the chance of FSS selecting an incorrect instance as seed is higher than that of ISS. Furthermore, in every iteration, ISS has three prior seeds to support the newly chosen one, which minimizes the chance of expanding seeds that are all incorrect.

Figure 4.3 shows that Random Walk is the most robust of the five rankers, followed by Bayesian Sets and Wrapper Length. While all rankers performed poorly with noisy seeds, only Random Walk improved (slightly at the 10^{th} iteration). Figure 4.4 shows that the performance of Random Walk increases monotonically when bootstrapping using ISS. It also shows that Random Walk has the best performance, followed by Bayesian Sets and Wrapper Length.

	S. Expansion		**Bootstrapping**	
	FSS	**ISS**	**FSS**	**ISS**
Avg. MAP @ 1^{st}	89.9	89.9	89.9	89.9
Avg. MAP @ 10^{th}	97.1	96.4	89.4	93.3
Relative Δ MAP	7.9%	7.2%	-0.6%	3.7%
Total # S. Seeds	20	11	2	2
Δ MAP / S. Seed	0.4%	0.7%	-0.3%	1.9%

Table 4.2: Performance (MAP) averaged over all five rankers for each iterative method.

	MAP @ 1^{st}	MAP @ 10^{th}	Δ MAP
Random Walk (RW)	90.3	93.9	3.9%
Bayesian Sets (BS)	90.3	93.7	3.7%
Wrapper Length (WL)	90.4	93.4	3.3%
Wrapper Frequency (WF)	89.8	93.1	3.7%
PageRank (PR)	88.8	92.4	4.0%
Average	89.9	93.3	3.7%

Table 4.3: Performance (MAP) of relative improvements (1^{st} to 10^{th} iteration) using bootstrapping with increasing seed size (ISS).

Figure 4.5, Figure 4.6, Figure 4.7, and Figure 4.8 show the performance of Random Walk, Bayesian Sets, Wrapper Length, and PageRank respectively, using four different configurations of iterative processes and seeding strategies. These figures illustrate that supervised expansion always performs better than bootstrapping. They also show that, among the four configurations, the best one is always FSS in supervised mode and the worst is also FSS but in bootstrapping mode. The two ISS configurations fall between the two FSS configurations, with supervised expansion being better than bootstrapping.

Table 4.2 summarizes the experimental results by averaging the performance of all rankers for each iterative process and seeding strategy. Since supervised seeds are difficult to obtain, we analyze the amount of relative improvement (in MAP) per supervised seed for each method. The table shows that supervised expansion using FSS gives the most improvement (7.9%) and produces the highest score (97.1%); however, it consumes 20 supervised seeds, resulting in a low improvement-per-seed ratio (0.4%). The one that gives the highest ratio is bootstrapping using ISS (1.9%), which shows

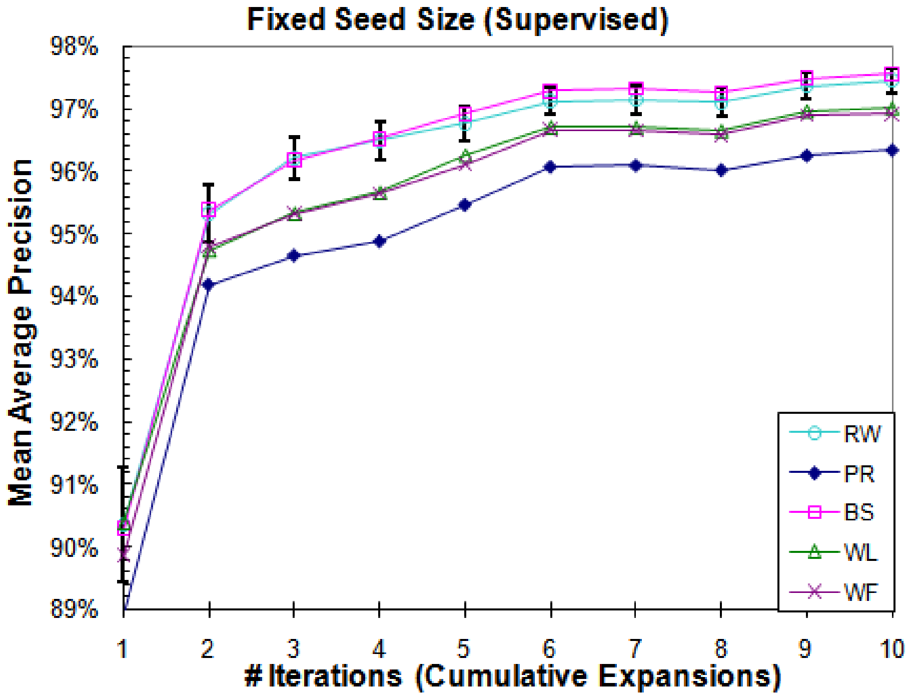


Figure 4.1: MAP of rankers using supervised expansion with fixed seed size (FSS).

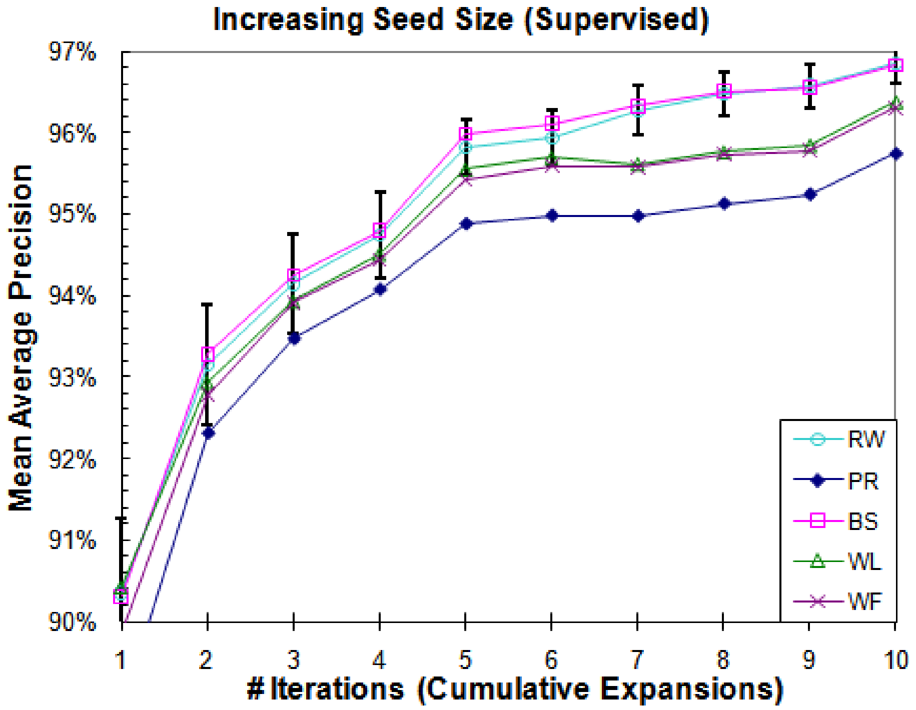


Figure 4.2: MAP of rankers using supervised expansion with increasing seed size (ISS).

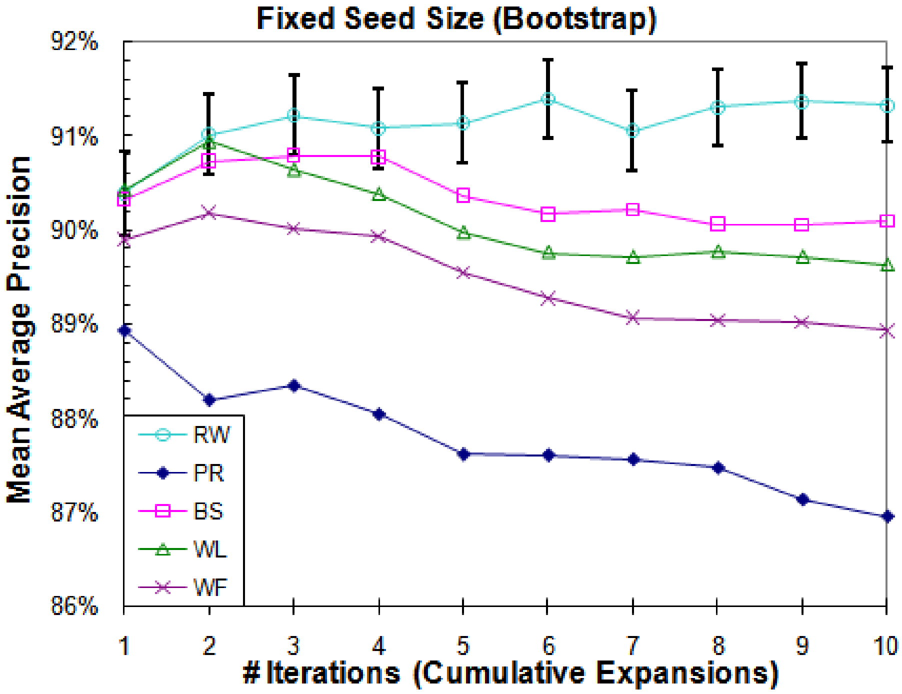


Figure 4.3: MAP of rankers using bootstrapping with fixed seed size (FSS).

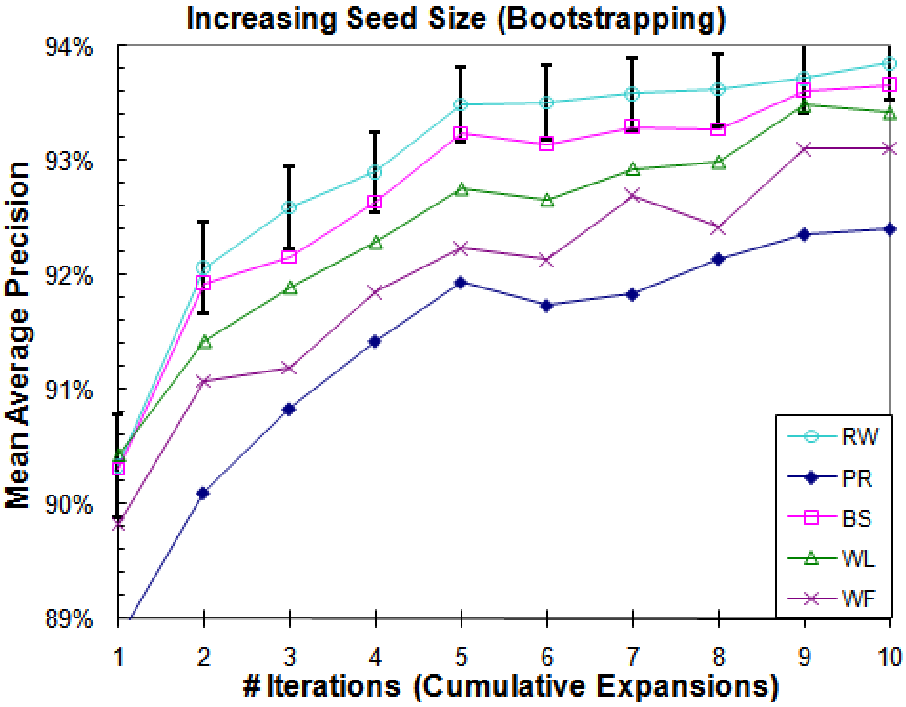


Figure 4.4: MAP of rankers using bootstrapping with increasing seed size (ISS).

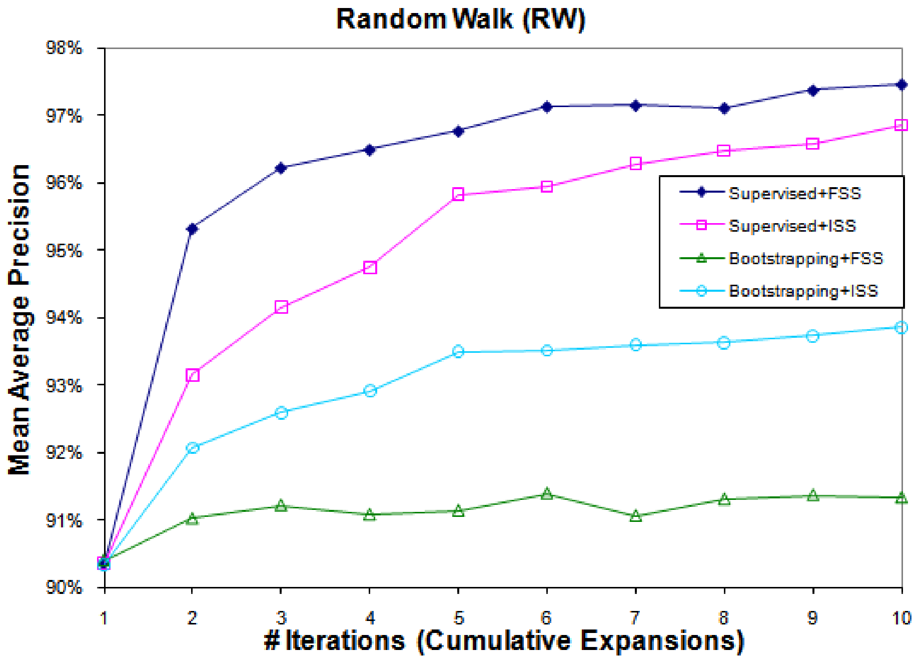


Figure 4.5: MAP of Random Walk using various iterative methods.

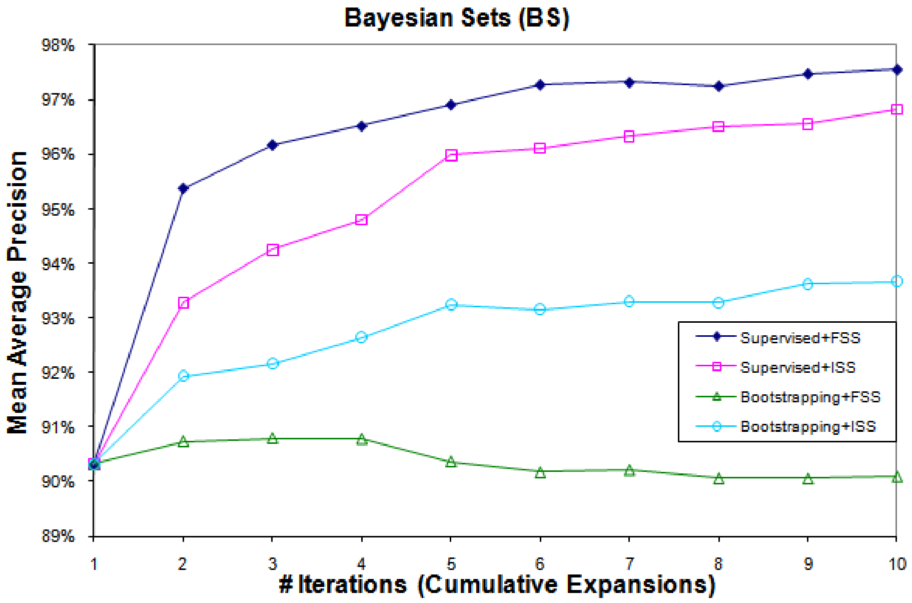


Figure 4.6: MAP of Bayesian Sets using various iterative methods.

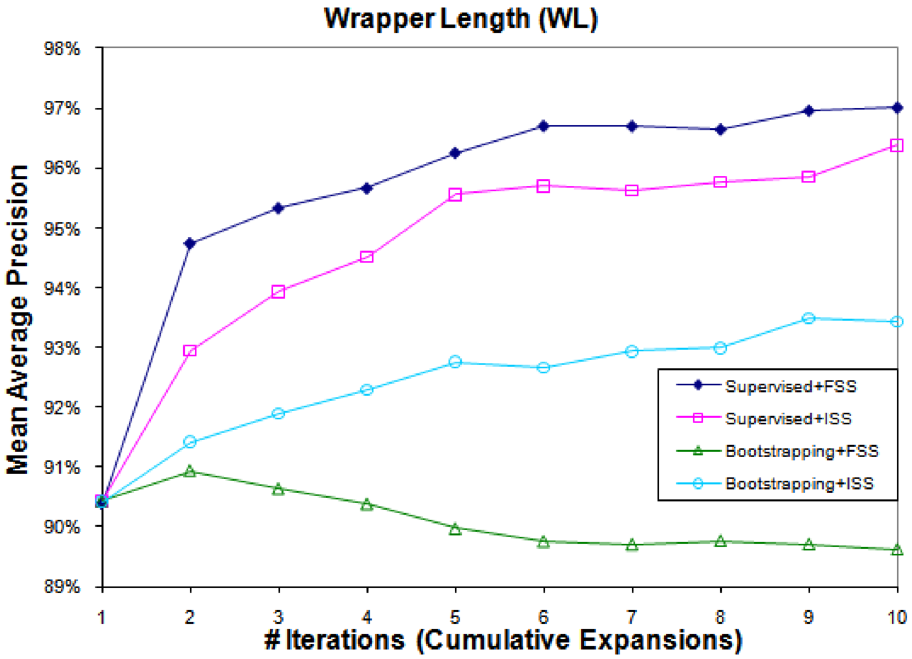


Figure 4.7: MAP of Wrapper Length using various iterative methods.

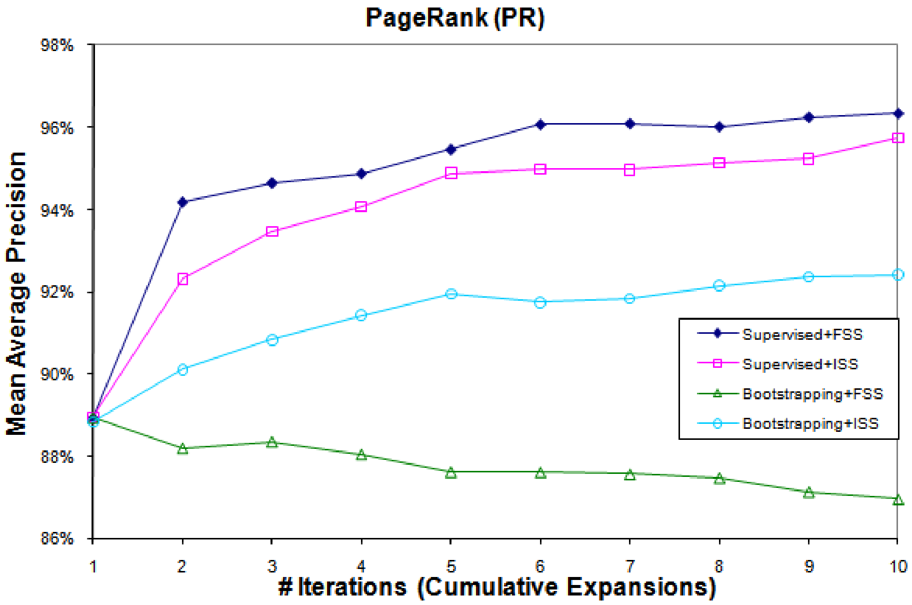


Figure 4.8: MAP of PageRank using various iterative methods.

	RW		**BS**		**WL**		**WF**		**PR**	
English	1^{st}	10^{th}	1^{st}	10^{th}	1^{st}	10^{th}	1^{st}	10^{th}	1^{st}	10^{th}
disney-movies	74.1	83.7	74.9	84.2	71.0	83.4	73.5	83.4	72.1	83.7
constellations	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
countries	98.4	98.8	98.3	98.8	97.9	98.6	98.0	98.8	97.1	98.6
mlb-teams	98.5	98.6	98.0	98.6	97.5	98.6	98.0	98.7	98.3	98.7
nba-teams	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
nfl-teams	100.0	100.0	100.0	89.6	100.0	100.0	100.0	100.0	100.0	100.0
car-makers	74.4	86.9	75.4	86.9	86.8	89.5	74.8	82.6	64.1	74.0
us-presidents	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
us-states	99.5	100.0	99.5	100.0	99.9	100.0	99.8	100.0	97.0	100.0
cmu-buildings	77.3	85.0	82.0	84.6	77.0	82.8	78.4	84.2	77.7	82.8
diseases	16.5	42.1	16.9	44.3	13.6	41.4	15.1	32.5	16.0	33.5
periodic-comets	78.1	85.4	77.8	86.1	80.7	84.9	79.1	86.8	72.3	71.8
Average	**84.7**	**90.0**	**85.2**	**89.4**	**85.4**	**89.9**	**84.7**	**88.9**	**82.9**	**86.9**
Chinese	1^{st}	10^{th}	1^{st}	10^{th}	1^{st}	10^{th}	1^{st}	10^{th}	1^{st}	10^{th}
disney-movies	80.5	88.5	79.9	88.2	81.3	88.7	78.4	87.0	78.1	86.7
constellations	99.7	100.0	99.5	100.0	99.7	100.0	99.7	99.9	99.7	100.0
countries	97.2	96.4	97.3	96.4	96.8	96.0	97.5	97.0	96.5	95.8
mlb-teams	99.8	99.9	99.8	99.8	99.6	99.7	99.8	99.8	99.9	99.9
nba-teams	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
nfl-teams	95.8	99.1	97.3	99.2	97.2	91.9	92.1	98.1	94.9	98.9
car-makers	88.4	96.2	87.8	96.1	85.9	96.0	86.7	96.1	86.1	95.6
us-presidents	94.7	96.5	94.8	96.3	94.3	95.8	93.5	95.7	94.1	95.2
us-states	99.2	100.0	99.4	100.0	98.6	100.0	99.1	100.0	99.2	100.0
china-dynasties	40.7	66.6	38.9	66.7	42.3	63.1	38.0	63.1	30.4	61.4
china-provinces	97.7	98.5	97.6	98.2	97.8	97.9	98.5	98.5	97.3	97.5
taiwan-cities	99.0	99.5	98.8	99.5	98.7	99.6	99.0	99.4	98.7	99.4
Average	**91.1**	**95.1**	**90.9**	**95.0**	**91.0**	**94.1**	**90.2**	**94.6**	**89.6**	**94.2**
Japanese	1^{st}	10^{th}	1^{st}	10^{th}	1^{st}	10^{th}	1^{st}	10^{th}	1^{st}	10^{th}
disney-movies	78.2	80.7	77.8	80.8	75.5	79.9	76.8	79.3	77.5	80.0
constellations	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
countries	97.2	98.0	96.7	98.0	97.7	97.7	97.3	97.0	95.6	97.6
mlb-teams	100.0	100.0	100.0	100.0	99.9	100.0	99.8	100.0	99.9	100.0
nba-teams	99.9	100.0	99.8	100.0	100.0	100.0	100.0	100.0	99.9	100.0
nfl-teams	99.3	100.0	99.2	99.9	99.2	99.9	99.7	100.0	99.2	100.0
car-makers	88.1	90.4	86.7	90.3	88.0	90.2	87.3	89.9	83.3	89.4
us-presidents	94.1	95.8	93.0	95.5	91.7	94.7	92.5	92.4	92.5	95.0
us-states	99.7	100.0	99.6	100.0	99.2	99.9	99.6	100.0	99.3	99.9
japan-emperors	99.2	99.2	99.2	99.2	99.0	99.2	97.6	98.4	99.2	99.2
japan-prime-mins	91.7	95.0	90.7	94.8	91.6	93.8	91.6	93.2	89.7	92.3
japan-provinces	95.3	100.0	94.6	100.0	96.6	100.0	92.4	100.0	92.4	99.6
Average	**95.2**	**96.6**	**94.8**	**96.5**	**94.9**	**96.3**	**94.6**	**95.9**	**94.0**	**96.1**
Overall Average	**90.3**	**93.9**	**90.3**	**93.7**	**90.4**	**93.4**	**89.8**	**93.1**	**88.8**	**92.4**

Table 4.4: Performance (MAP) of various rankers at 1^{st} and 10^{th} iteration using bootstrapping with increasing seed size (ISS).

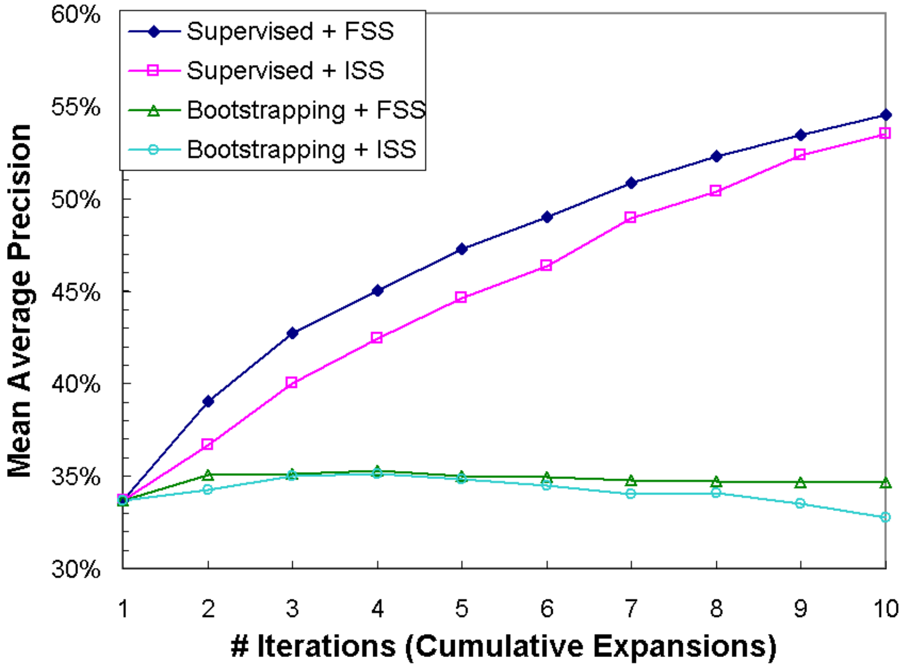


Figure 4.9: MAP of Google Sets using various iterative methods.

that it is an effective bootstrapping method. Examining more closely at this particular method, Table 4.4 presents a very detailed set of results for each ranker using this method, while Table 4.3 summarizes and shows that each ranker's performance using this method gives an relative improvement of about 3.7% in MAP. The results show that this method is effective for every ranker, even for the ranker PageRank which has the worst performance in every experiment. The results also indicate that Random Walk is able to reach a MAP of nearly 94% at the 10^{th} iteration using only two supervised seeds. This performance is even better than the MAP of 93% achieved by SEAL using the same number of web pages (i.e., 100) but more supervised seeds (i.e., 3) as presented in Table 2.9.

The graphs show that Wrapper Length is also an effective ranking algorithm. It is comparable to the baseline Wrapper Frequency in supervised mode and is always better in the bootstrap mode. We want to emphasize that Wrapper Length is a very simple and light-weight algorithm: unlike graph-based rankers, the memory space needed is proportional only to the number of extracted candidate instances. Furthermore, the results suggest that if many supervised seeds are available, then supervised expansion using FSS is more preferable, but if only a few are available, then bootstrapping using ISS is more preferable. In terms of rankers, both Bayesian Sets and Random Walk are good choices for supervised expansion, and Random Walk is the best choice for bootstrapping.

Since Google Sets is publicly available online, it is natural for us to examine the performance of Google Sets using the iterative set expansion methods presented in this paper. Google Sets returns a list of items without their confidence scores; thus, we assume each item has a weight of $1 - \frac{r-1}{|L|}$, where r is the rank and L is the ranked list. This gives the first item a weight of one and the last item a weight close to zero. When we accumulate statistics, we simply sum up the weights for every item. Figure 4.9 shows the performance of iterative set expansion using Google Sets. For the supervised expansion, Google Sets shows a monotonic increase of performance for both FSS and ISS; however, it shows no improvement for either bootstrapping method. As mentioned earlier, bootstrapping is sensitive to a system's performance. Since Google Sets has a base performance (the performance at the first iteration) of only about 34% in MAP (SEAL has 90%), its top ranked items are often incorrect, causing errors to propagate easily and worsening the results.

4.5 Summary

In this chapter, we have presented a system called Iterative SEAL (iSEAL) to examine various iterative processes and seeding strategies using different rankers for the problem of set expansion. We have shown that the performance of SEAL can improve monotonically if we bootstrap the results using ISS and rank the results using Random Walk with Restart. By using this method and only two seeds, the final result (94%) is even better than that of using three supervised seeds (with same amount of web pages) as presented in Section 2.5 (93%). We have also shown that in supervised mode, Random Walk is comparable to the best ranker (Bayesian Sets), but in bootstrap mode, Random Walk is the best due to its robustness to noisy seeds. We have also presented a simple and light-weight ranker, Wrapper Length, that shows good performance in most experiments. From the experimental results, we have learned that if there are many supervised seeds, then the best way to improve performance is by supervised expansion using FSS. However, if there are only a few supervised seeds, then the best way is by bootstrapping using ISS. In the future, we would like to investigate whether or not bootstrapping with ISS will continue to improve the performance of Random Walk monotonically after ten iterations. In addition, we would like to examine in detail the reason why Random Walk performs well while PageRank performs poorly compared to other rankers.

Chapter 5

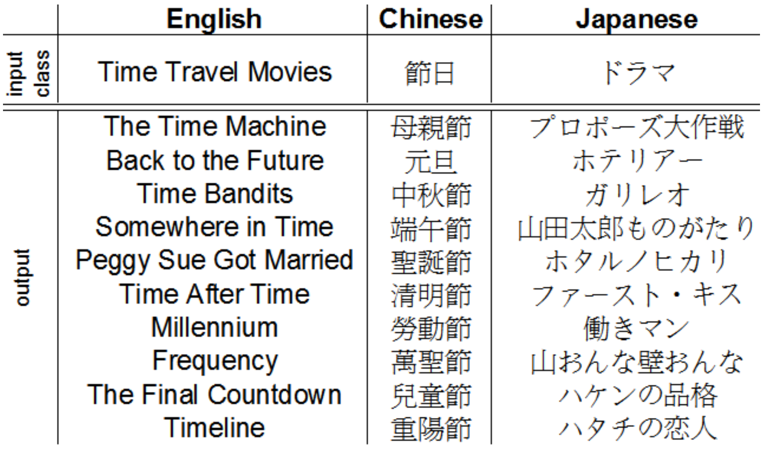
Automatic Set Instance Acquisition

5.1 Introduction

An important and well-studied problem is the production of semantic lexicons for classes of interest; that is, the generation of all instances of a set (e.g., “apple”, “orange”, “banana”) given a name of that set (e.g., “fruits”). This task is often addressed by linguistically analyzing very large collections of text [15, 17, 20, 26, 32], using hand-constructed or machine-learned shallow linguistic patterns to detect *hyponym* instances. A hyponym is a word or phrase whose semantic range is included within that of another word. The opposite of hyponym is *hypernym.* For example, apple is a hyponym of fruits and fruits is a hypernym of apple because apple is a (kind of) fruits. More formally, x is a hyponym of y and y is a hypernym of x if x is a (kind of) y.

In this chapter, we evaluate a novel approach to this problem, embodied in a system called ASIA[1] (Automatic Set Instance Acquirer). ASIA takes a semantic class name as input (e.g., “car makers”) and automatically outputs instances (e.g., “ford”, “nissan”, “toyota”). Unlike prior methods, ASIA makes heavy use of a web-based set expansion system, specifically, the SEAL system. SEAL has been extended to be robust to errors in its initial set of seeds, as described in Chapter 3, and to use bootstrapping to iteratively improve its performance, as described in Chapter 4. These extensions allow ASIA to extract instances of sets from the Web, as follows. First, given a semantic class

[1] `http://rcwang.com/asia`



	English	Chinese	Japanese
input class	Time Travel Movies	節日	ドラマ
output	The Time Machine	母親節	プロポーズ大作戦
	Back to the Future	元旦	ホテリアー
	Time Bandits	中秋節	ガリレオ
	Somewhere in Time	端午節	山田太郎ものがたり
	Peggy Sue Got Married	聖誕節	ホタルノヒカリ
	Time After Time	清明節	ファースト・キス
	Millennium	勞動節	働きマン
	Frequency	萬聖節	山おんな壁おんな
	The Final Countdown	兒童節	ハケンの品格
	Timeline	重陽節	ハタチの恋人

Figure 5.1: Examples of ASIA's input and output. Input class for Chinese is "holidays" and for Japanese is "dramas".

name (e.g., "fruits"), ASIA uses a small set of language-dependent *hyponym patterns* (e.g., "fruits such as __") to find a large but noisy set of seed instances. Second, ASIA uses the extended version of SEAL to expand the noisy set of seeds. Figure 5.1 illustrates some real example inputs and outputs of ASIA in English, Chinese, and Japanese (more examples are shown in Table 5.8 and Figure 5.6 at the end of this chapter).

ASIA's approach is motivated by the conjecture that for many natural classes, the amount of information available in semi-structured documents on the Web is much larger than the amount of information available in free-text documents; hence, it is natural to attempt to augment search for set instances in free-text with semi-structured document analysis. We show that ASIA performs extremely well experimentally. On the 36 benchmarks presented in Section 2.5.4, which are relatively small closed sets (e.g., countries, constellations, NBA teams), ASIA has excellent performance for both recall and precision. On four additional English-language benchmark problems (US states, countries, singers, and common fish), we compare to recent work by Kozareva, Riloff, and Hovy [20], and show comparable or better performance on each of these benchmarks; this is notable because ASIA requires less information than the work of Kozareva *et al* (their system requires a concept name and a seed). We also compare ASIA on twelve additional benchmarks from the extended Wordnet 2.1 produced by Snow *et al* [38], and show that for these twelve sets, ASIA produces more than five times as many set instances with much higher precision (98% versus 70%).

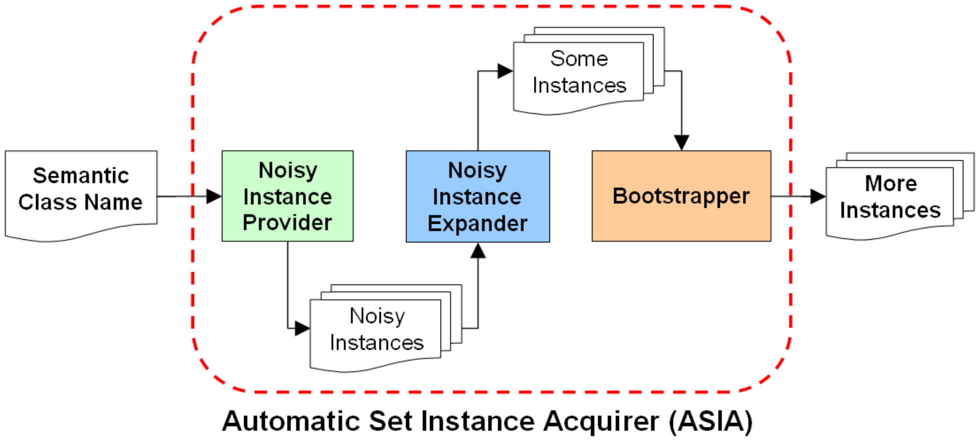
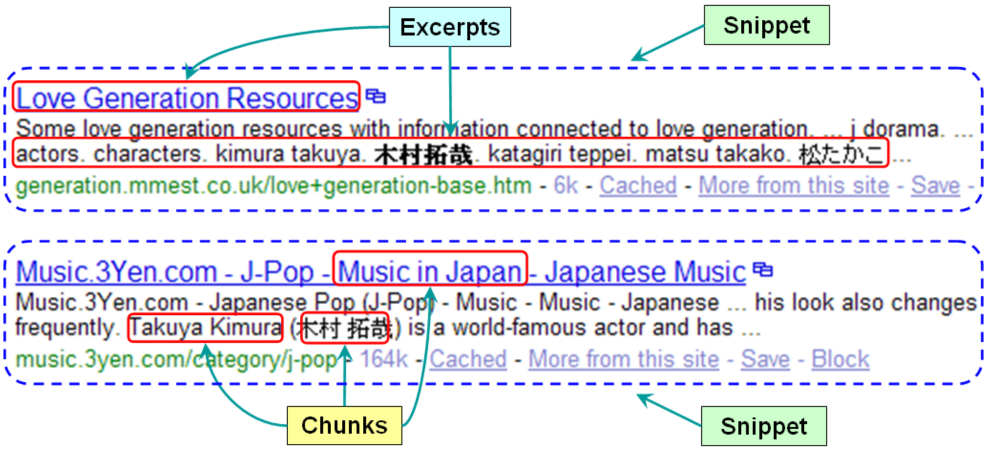


Figure 5.2: Flow chart of the ASIA system.



Love Generation Resources
Some love generation resources with information connected to love generation. ... j dorama. ...
actors. characters. kimura takuya. 木村拓哉. katagiri teppei. matsu takako. 松たかこ ...
generation.mmest.co.uk/love+generation-base.htm - 6k - Cached - More from this site - Save -

Music.3Yen.com - J-Pop - Music in Japan - Japanese Music
Music.3Yen.com - Japanese Pop (J-Pop) - Music - Music - Japanese ... his look also changes frequently. Takuya Kimura (木村 拓哉) is a world-famous actor and has ...
music.3yen.com/category/j-pop - 164k - Cached - More from this site - Save - Block

Figure 5.3: Examples of snippets, excerpts, and chunks.

Another advantage of ASIA's approach is that it is nearly language-independent: since the underlying set-expansion tools are language-independent, all that is needed to support a new target language is a new set of hyponym patterns for that language. In this chapter, we present experimental results for Chinese and Japanese, as well as English, to demonstrate this language-independence.

This chapter is organized as follows. Section 5.2 describes our related work, and Section 5.3 explains our proposed approach for ASIA. Section 5.4 presents the details of our experiments, as well as the experimental results. A comparison of results to prior work are illustrated and discussed in Section 5.5. We summarize this chapter in Section 5.6.

5.2 Related Work

There has been a significant amount of research done in the area of semantic class learning (aka lexical acquisition, lexicon induction, hyponym extraction, or open-domain information extraction). However, to the best of our knowledge, there is not a system that can perform set instance extraction in multiple languages given only the *name* of the set (e.g., Disney movies).

Hearst [17] presented an approach that utilizes hyponym patterns for extracting candidate instances. The approach presented in Section 5.3.1 is based on this work, except that we extended it to two other languages: Chinese and Japanese. Pantel *et al.* [32] presented an algorithm for automatically inducing names for semantic classes and for finding their instances by using "concept signatures" (statistics on co-occuring instances). Pasca [26] presented a method for acquiring named entities in arbitrary categories using lexico-syntactic extraction patterns. Etzioni *et al.* [15] presented the KnowItAll system that also utilizes hyponym patterns to extract class instances from the Web. All the systems mentioned above rely on either capitalization, English tokenizers, English part-of-speech taggers, and/or English parsers; hence, they are language-dependent.

Kozareva *et al* [20] illustrated an approach that uses a single hyponym pattern combined with graph structures to learn semantic class from the Web. Section 5.5.1 shows that our approach is competitive experimentally; however, their system requires more information, as it uses the name of the semantic set and a seed instance.

Paşca [27, 28] illustrated a set expansion approach that extracts instances from Web search queries given a set of input seed instances. This approach is similar in flavor to SEAL but, addresses a different task from that addressed here: for ASIA the user provides no seeds, but instead provides the *name* of the set being expanded. We compare to Paşca's system in Section 5.5.2.

Snow *et al.* [38] use known hypernym/hyponym pairs to generate training data for a machine-learning system, which then learns many lexico-syntactic patterns. The patterns learned are based on English-language dependency parsing. We compare to Snow *et al*'s results in Section 5.5.4.

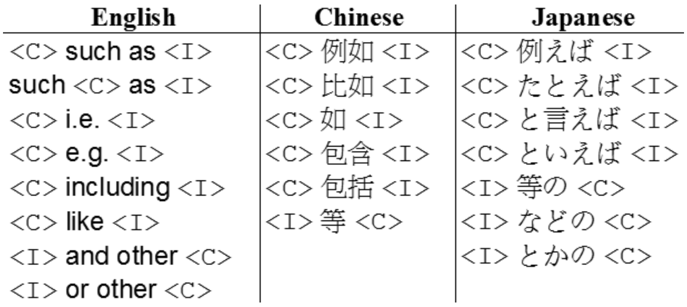
5.3 Proposed Approach

ASIA is composed of three main components: the *Noisy Instance Provider*, the *Noisy Instance Expander*, and the *Bootstrapper*. Given a semantic class name, the Provider extracts a initial set of noisy candidate instances using hand-coded patterns, and ranks the instances by using a simple ranking model. The Expander expands and ranks the instances using evidence from semi-structured web documents, such that irrelevant ones are ranked lower in the list. The Bootstrapper enhances the quality and completeness of the ranked list by using an unsupervised iterative technique. The architecture of ASIA is shown in Figure 5.2. Note that the Expander and Bootstrapper rely on SEAL to accomplish their goals. In this section, we first describe the Noisy Instance Provider, then the Noisy Instance Expander, followed by the Bootstrapper.

5.3.1 Noisy Instance Provider

The Noisy Instance Provider extracts candidate instances from free text (i.e., search engine results) using the methods presented in Hearst's early work [17]. Hearst exploited several patterns for identifying hyponymy relation (e.g., "such author as Shakespeare") that are implemented in many current state-of-the-art systems [15, 20, 26, 32]. However, unlike all of those systems, ASIA does not use any NLP tool (e.g., parts-of-speech tagger, parser) or rely on capitalization for extracting candidates (since we wanted ASIA to be as language-independent as possible). This leads to sets of instances that are very noisy; however, we will show that set expansion and re-ranking can improve the initial sets dramatically. Below, we will refer to the initial set of noisy instances extracted by the Provider as the *initial* set.

In more detail, the Provider first constructs a few queries of hyponym phrases by using a semantic class name and a set of pre-defined hyponym patterns, which are shown in Figure 5.4. For example, when given the semantic class name "periodic comets", the queries are as follows:



English	Chinese	Japanese
<C> such as <I>	<C> 例如 <I>	<C> 例えば <I>
such <C> as <I>	<C> 比如 <I>	<C> たとえば <I>
<C> i.e. <I>	<C> 如 <I>	<C> と言えば <I>
<C> e.g. <I>	<C> 包含 <I>	<C> といえば <I>
<C> including <I>	<C> 包括 <I>	<I> 等の <C>
<C> like <I>	<I> 等 <C>	<I> などの <C>
<I> and other <C>		<I> とかの <C>
<I> or other <C>		

Figure 5.4: Hyponym patterns in English, Chinese, and Japanese. In each pattern, `<C>` is a placeholder for the semantic class name and `<I>` is a placeholder for a list of instances.

1. “periodic comets such as”
2. “such periodic comets as”
3. “periodic comets i.e.”
4. “periodic comets e.g.”
5. “periodic comets including”
6. “periodic comets like”
7. “and other periodic comets”
8. “or other periodic comets”

Below, we refer to each of the search results as a *snippet*, each of the phrases delimited by “. . . ” in a snippet as an *excerpt*, and each sequence of characters bounded by punctuation marks or the beginning and end of an excerpt as a *chunk*. Figure 5.3 illustrates examples of snippets, excerpts, and chunks.

For every query, the Provider retrieves two hundred snippets from Yahoo!, and parses each snippet into a structured format containing elements such as the title, page URL, and list of excerpts (a snippet often contains multiple continuous excerpts from its web page). For each excerpt, the Provider uses a regular-expression pattern, which specifies the allowable characters in a particular language, to extract all chunks in that language. These chunks will then be used as candidate instances. Lastly, the Provider ranks each candidate instance c based on its weight assigned by the simple ranking model presented below:

$$weight(c) = \frac{hf(c, \mathbf{H})}{|\mathbf{H}|} \times \frac{sf(c, \mathbf{S})}{|\mathbf{S}|} \times \frac{ef(c, \mathbf{E})}{|\mathbf{E}|} \times \frac{wcf(c, \mathbf{E})}{|\mathbf{C}|}$$

where $\mathbf{H}$ is the set of hyponym patterns, $\mathbf{S}$ the set of snippets, $\mathbf{E}$ the set of excerpts, and $\mathbf{C}$ the set of chunks. The function $hf(c, \mathbf{H})$ is the hyponym pattern frequency of c (i.e., the number of hyponyn patterns inducing c), the function $sf(c, \mathbf{S})$ is the snippet frequency of c (i.e., the number of snippets containing c), and $ef(c, \mathbf{E})$ is the excerpt frequency of c (i.e., the number of excerpts containing c). Furthermore, $wcf(c, \mathbf{E})$ is the weighted chunk frequency of c, which is defined as follows:

$$wcf(c, \mathbf{E}) = \sum_{e \in \mathbf{E}} \sum_{c \in e} \frac{1}{dist(c, e) + 1}$$

where $dist(c, e)$ is the number of characters between c and the hyponym phrase in excerpt e. This model weights every occurrence of c based on the assumption that chunks closer to a hyponym phrase are usually more important than those further away. It also heavily rewards frequency, as our assumption is that the most common instances will be more useful as seeds for SEAL.

As shown in Figure 5.4, there are two types of hyponym patterns: The first type are the ones that require the class name C to precede its instance I (e.g., C such as I), and the second type are the opposite ones (e.g., I and other C). In order to reduce irrelevant chunks, when excerpts were extracted, the Provider drops all characters preceding the hyponym phrase in excerpts that contain the first type, and also drops all characters following the hyponym phrase in excerpts that contain the second type. For some semantic class names (e.g., “cmu buildings”), there are no web documents containing any of the hyponym-phrase queries that were constructed using the name. In this case, the Provider turns to a *back-off* strategy which simply treats the semantic class name as the hyponym phrase, and then extracts and ranks all chunks co-occurring with the class name in the excerpts.

Below, we show the top 20 chunks extracted and ranked by the Provider when given the input semantic class name “periodic comets”. Note that the ones that exist in our evaluation dataset are marked with a ‘+’. Although the entire ranked list is very noisy and has a low mean average precision of mere 2%, our noise-resistant set expansion approach, described in the next section, will improve the quality of this list.

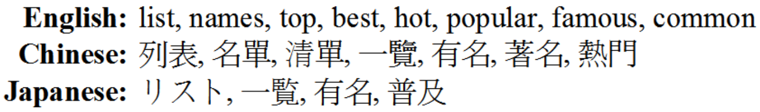
1. +Halley
2. −Comet Halley are seen to orbit on a regular path
3. −Halley’s Comet once followed a much
4. +Wild 2
5. −this one are doomed to disintegrate

6. −die
7. −but without showing a cometary coma
8. +Tempel-Swift-LINEAR
9. −Comet Encke
10. +Neujmin
11. −but their periods are extremely long
12. −studying comet P
13. +Shoemaker-LINEAR
14. +Tempel 1
15. −McNaught P
16. −this one are doomed to
17. −R2
18. −Lagerkvist
19. −originate
20. −which moves in an elongated ellipse

5.3.2 Noisy Instance Expander

Chapter 3 illustrated that it is feasible to perform set expansion on noisy input seeds. It also showed that the noisy output of any Question Answering system for list questions can be improved by using a noise-resistant version of SEAL (An example of a list question is "Who were the husbands of Heddy Lamar?"). However, unlike in the Question Answering task, here we have an additional piece of information, that is, the semantic class name of the desired list. This information allows us to use a more efficient approach for the Expander to expand noisy seeds.

The Expander performs set expansion on a collection of two hundred web pages. This collection is fetched by sending one single query to Google and Yahoo! (each returns one hundred pages); instead of $\binom{n}{2}$ queries as done by the noise-resistant SEAL. The query is composed of the input class name and a concatenated string of *list words* (i.e, words that often co-occur with list-containing pages) for discovering web pages that might contain lists of the input class. The table below shows the set of list words we use in our experiments.



English:	list, names, top, best, hot, popular, famous, common
Chinese:	列表, 名單, 清單, 一覽, 有名, 著名, 熱門
Japanese:	リスト, 一覧, 有名, 普及

For example, when given the semantic class name "periodic comets", the search query would be the following:

"periodic comets" (list OR names OR top OR best OR hot OR popular OR famous OR common)

The Expander expands the top 20 instances in the initial set, using a variant of the noise-resistant set expansion approach described in Section 3.3.2. The previously-described approach, referred to as NE1, requires the contexts to bracket at least one instance of *a minimum of two seeds* per web page. However, this variant, referred to as NE2, requires the common contexts that bracket the *largest number of unique seeds* to be as long as possible per web page. More specifically, given some seeds **S** and a document d, finds all wrappers w such that all the following are true:

1. w covers seeds S_w in d where $S_w \in \mathbf{S}$ and $|S_w| \geq 2$
2. $\neg\exists w'$: w' covers seeds $S_{w'}$ in d where $|S_{w'}| > |S_w|$
3. $\neg\exists w'$: w' covers seeds S_w in d where $|w'| > |w|$

This approach favors the coverage of wrappers; thus, although we observed that the length of the constructed wrappers are generally shorter, we will show later that they are actually more effective. The idea is based on the assumption that irrelevant instances usually do not have common contexts; whereas relevant ones do. Both NE1 and NE2 were implemented using the trie-based method described in Section 2.3.2. The ranking method used by the Expander is Random Walk with Restart [43], described in Section 2.4.3, which was also shown to be the most robust to noisy seeds in Figure 4.4. In our experimental results, we show that the performance of NE2 is better than NE1 on our evaluation datasets.

Below, we show the top 20 instances produced by the Expander when using as seeds the top 20 instances extracted by the Extractor on the input class name "periodic comets". Note that the ones that exist in our evaluation dataset are marked with a '+', and the mean average precision of the entire ranked list is 74% (compared to the previous 2%).

1. +Halley
2. +Encke
3. +Wild 2
4. +Borrelly
5. +Tempel 1
6. +Kohoutek
7. +Churyumov-Gerasimenko

11. +Biela
12. −Hale-Bopp
13. +Grigg-Skjellerup
14. +Wirtanen
15. −Hyakutake
16. +Crommelin
17. +Holmes

8. +Tuttle
9. +Giacobini-Zinner
10. +Whipple
18. +Tempel-Tuttle
19. +Chiron
20. +Peters-Hartley

5.3.3 Bootstrapper

Bootstrapping [15, 19, 25] is an unsupervised iterative process in which a system continuously consumes its own outputs to improve its own performance. Chapter 4 showed that it is feasible to bootstrap the results of set expansion to improve the quality of a list. The chapter introduces an iterative version of SEAL called iSEAL, which expands a list in multiple iterations. In each iteration, iSEAL expands a few candidates extracted in previous iterations and aggregates statistics. The Bootstrapper utilizes iSEAL to further improve the quality of the list returned by the Expander.

The Bootstrapper configures iSEAL to bootstrap with increasing seed size, as detailed in Section 4.2.2.2. Recall that this seeding strategy starts with an empty set of previously *used* seeds. However, since we want ASIA to run as fast as possible, *used* was initialized to contain instances extracted by both the Extractor and the Expander because we observed that those overlapping instances have high precision. Also, this method starts by expanding four randomly-chosen instances from *used*, rather than two (supervised) seeds. For every successive iteration, it expands three randomly-chosen instances from *used* and one top (unsupervised) instance from the last iteration. However, there is one limitation on this top instance – it must also be in the initial set extracted by the Extractor in order to minimize the chance of selecting irrelevant instances to be used as seeds.

At every iteration, this bootstrapping method retrieves one hundred pages from Yahoo! by submitting a four-seed query, which is the number of seeds that maximizes set expansion performance, as shown in Table 4.1. In order to keep the overall runtime minimal, the iterative expansion process terminates when all possible seed combinations have been consumed or five iterations have been reached, whichever comes first. Notice that from iteration to iteration, statistics are aggregated by growing the graph originally constructed by the Expander. We perform Random Walk on this graph to determine the final ranking of the extracted instances.

When given the semantic class name "periodic comets", the *used* seed set was initialized to contain the following intersected instances: *Halley*, *Wild 2*, *Encke*, and

English Dataset	Class Name	Chinese Dataset	Class Name	Japanese Dataset	Class Name
disney-movies	Disney Movies	disney-movies	迪士尼動畫	disney-movies	ディズニー映画
constellations	Constellations	constellations	星座	constellations	星座
countries	Countries	countries	國家	countries	国
mlb-teams	MLB Teams	mlb-teams	MLB 球隊	mlb-teams	MLB チーム
nba-teams	NBA Teams	nba-teams	NBA 球隊	nba-teams	NBA チーム
nfl-teams	NFL Teams	nfl-teams	NFL 球隊	nfl-teams	NFL チーム
car-makers	Car Makers	car-makers	車廠	car-makers	自動車メーカー
us-presidents	US Presidents	us-presidents	美國總統	us-presidents	アメリカ大統領
us-states	US States	us-states	州	us-states	アメリカの州
cmu-buildings	CMU Buildings	china-dynasties	朝代	japan-emperors	天皇
diseases	Diseases	china-provinces	省	japan-prime-mins	総理大臣
periodic-comets	Periodic Comets	taiwan-cities	縣市	japan-provinces	県

Figure 5.5: The 36 datasets and their semantic class names used as inputs to ASIA in our experiments.

Kohoutek, which are all correct periodic comets. After bootstrapping, the quality of the ranked list improved from 74% to 86% in mean average precision.

5.4 Experimental Results

We evaluate our proposed approach using the 36 evaluation datasets described in Section 2.5.4 and the evaluation metric – Mean Average Precision presented in Section 2.5.5. For each semantic class in our datasets, we take its corresponding class name from the table shown in Figure 5.5 and provide the name to the Provider. The Provider then produces a noisy list of candidate instances, which is then expanded by the Expander and further improved by the Bootstrapper.

We present our experimental results in Table 5.3. As illustrated, NE2 performs better than NE1 on our datasets, scoring 91% in MAP; whereas NE1 scored 84%. Although the Provider performs badly, NE2 substantially improves the quality of the initial list. On average, NE2 improves the performance of the Provider from 41% to 87% for English, 26% to 92% for Chinese, and 17% to 95% for Japanese. In addition, the results illustrate that the Bootstrapper is also effective. It improves the performance of NE1 from 84% to 88% in overall average and the performance of NE2 from 91% to 93%. It even directly improves the performance of the Provider from 41% to 79% for English, 26% to 40% for Chinese, and 17% to 35% for Japanese.

The simple *back-off* strategy seems to be effective as well. There are two datasets

		NP	NP	NP	NP	NP
English Dataset	**NP**	**+BS**	**+NE1**	**+NE1+BS**	**+NE2**	**+NE2+BS**
disney-movies	0.24	0.82	0.63	0.40	0.82	0.87
constellations	0.33	1.00	1.00	1.00	1.00	1.00
countries	0.64	0.97	0.97	0.97	0.99	0.98
mlb-teams	0.65	1.00	1.00	1.00	1.00	1.00
nba-teams	0.51	0.88	0.88	1.00	1.00	1.00
nfl-teams	0.69	1.00	1.00	1.00	1.00	1.00
car-makers	0.45	0.98	0.94	0.96	0.98	0.98
us-presidents	0.38	1.00	0.92	1.00	1.00	1.00
us-states	0.89	1.00	1.00	1.00	1.00	1.00
cmu-buildings	0.00	0.00	0.00	0.00	0.00	0.00
diseases	0.10	0.84	0.72	0.72	0.89	0.87
periodic-comets	0.02	0.00	0.38	0.38	0.74	0.86
Average	0.41	0.79	0.79	0.79	0.87	0.88
Chinese Dataset	**NP**	**+BS**	**+NE1**	**+NE1+BS**	**+NE2**	**+NE2+BS**
disney-movies	0.09	0.00	0.65	0.90	0.89	0.95
constellations	0.19	0.20	0.90	0.77	0.88	0.90
countries	0.36	0.86	0.56	0.93	0.82	0.89
mlb-teams	0.02	0.00	1.00	1.00	1.00	1.00
nba-teams	0.16	0.00	0.93	1.00	0.94	1.00
nfl-teams	0.05	0.00	0.93	0.68	0.98	0.96
car-makers	0.09	0.80	0.51	0.88	0.86	0.95
us-presidents	0.20	0.00	0.88	0.95	0.94	0.94
us-states	0.66	0.95	0.99	1.00	1.00	1.00
china-dynasties	0.06	0.00	0.80	0.78	0.74	0.86
china-provinces	0.72	0.94	0.89	0.98	1.00	1.00
taiwan-cities	0.50	1.00	1.00	1.00	1.00	1.00
Average	0.26	0.40	0.84	0.91	0.92	0.95
Japanese Dataset	**NP**	**+BS**	**+NE1**	**+NE1+BS**	**+NE2**	**+NE2+BS**
disney-movies	0.12	0.65	0.51	0.78	0.72	0.74
constellations	0.10	0.01	1.00	0.95	1.00	1.00
countries	0.28	0.95	0.97	0.98	0.98	0.98
mlb-teams	0.07	0.99	0.97	1.00	0.97	1.00
nba-teams	0.06	0.00	1.00	1.00	0.99	1.00
nfl-teams	0.04	0.00	0.92	0.92	0.92	0.92
car-makers	0.33	0.67	0.62	0.84	0.88	0.90
us-presidents	0.08	0.00	0.87	0.92	0.90	0.90
us-states	0.29	0.14	1.00	1.00	1.00	1.00
japan-emperors	0.07	0.00	1.00	1.00	1.00	1.00
japan-prime-mins	0.26	0.38	0.94	0.97	0.99	0.99
japan-provinces	0.37	0.42	1.00	1.00	1.00	1.00
Average	0.17	0.35	0.90	0.95	0.95	0.95
Overall Average	**0.28**	**0.51**	**0.84**	**0.88**	**0.91**	**0.93**

Table 5.3: Performance (MAP) of various system configurations for each dataset, where NP is the Noisy Instance Provider and BS is the Bootstrapper. NE1 is the Noisy Instance Expander that implements the noise-resistant expansion approach described in Section 3.3.2, and NE2 is a variant of NE1.

(i.e., English "cmu-buildings" and Japanese "nfl-teams") for which their hyponym phrases return no web documents. For those datasets, ASIA automatically uses the *back-off* strategy described in Section 5.3.1. This strategy failed for "cmu-buildings" but works for "nfl-teams"; it scored 4% on this dataset, which is further improved to 92%.

5.5 Comparison to Prior Work

In this section, we compare the best-configured ASIA (NP+NE2+BS) against five other published works.

5.5.1 Kozareva *et al.*, 2008

Kozareva *et al.* [20] present an approach to weakly supervised semantic class learning from search engine snippets, using a single hyponym pattern (i.e., CLASS_NAME *such as* INSTANCE *and* ⋆) combined with graph structures. They developed algorithms that begin with just a class name and one seed instance and then automatically generate a ranked list of new instances. In their extractor, for proper instance names, they extract all capitalized words that immediately follow their learned patterns, but for common noun instances, they extract just one word if it is not capitalized.

They report results on four semantic classes: *US states*, *countries*, *singers*, and *common fish*. We evaluated our results on those four classes automatically except for *singers*, which we evaluated manually because we do not have a complete list of all singers in the world. We used a list of common fish from the Wikipedia[1], which is also used by Kozareva *et al.* The results indicate that ASIA outperforms theirs for all four datasets that they reported. Note that the input to their system is a semantic class name plus one seed instance; whereas, the input to ASIA is only the class name. In terms of system runtime, for each semantic class, they reported that their extraction process usually finished overnight; however, ASIA usually finished within a minute running on a single CPU machine.

If ASIA was provided with a semantic class name and a seed instance, like the system in Kozareva *et al.*, then ASIA could utilize the seed instance in two ways – a simple and a complicated way. The simple way is for the Expander to filter out

[1] `http://en.wikipedia.org/wiki/List_of_fish_common_names`

N	Kozareva	ASIA	N	Kozareva	ASIA
US States			Countries		
25	1.00	1.00	50	1.00	1.00
50	1.00	1.00	100	1.00	1.00
64	0.78	0.78	150	1.00	1.00
			200	0.90	0.93
			300	0.61	0.67
			323	0.57	0.62
Singers			Common Fish		
10	1.00	1.00	10	1.00	1.00
25	1.00	1.00	25	1.00	1.00
50	0.97	1.00	50	1.00	1.00
75	0.96	1.00	75	0.93	1.00
100	0.96	1.00	100	0.84	1.00
150	0.95	0.97	116	0.80	1.00
180	0.91	0.96			

Table 5.4: Set instance extraction performance compared to Kozareva *et al.* We report our precision for all semantic classes and at the same ranks reported in their work.

wrappers that did not extract the seed instance. The complicated way is to modify the wrapper construction algorithm in the Expander, such that every newly extracted instance must also occur in the same (left and right) contexts as the seed instance.

5.5.2 Paşca, 2007b

Paşca [28] present a set expansion approach from web search queries given a small set of seed named entities, which is the same interface as SEAL. Nevertheless, we will compare his system against ASIA, which requires less information than SEAL and their approach. The show that search queries are highly valuable resource for web-based named entity discovery. Their extraction method consists of five stages: identification of query templates that match the seed instances; identification of candidate instances; internal representation of candidate instances and seed instances; and instance ranking. The input to their experiments is a random sample of around 50 million unique, fully-anonymized queries in English submitted by users to the Google search engine in 2006.

We compare our system ASIA against Paşca and present comparison results in Table 5.5. There are 10 semantic classes in his evaluation datasets. We input to ASIA each of the class names shown in the left-most column of Table 5.5, and we manually evaluated every output instance from ASIA for each class (except for *countries*, which was done automatically). The results show that ASIA has higher precision at each rank reported in his work. For instance, ASIA has perfect precision at rank 50; whereas, his system achieves 97%. At rank 250, ASIA achieves an average of 93% precision; whereas, his system achieves 80%.

		Precision @				
Class Name	**System**	**25**	**50**	**100**	**150**	**250**
Cities	Pasca	1.00	0.96	0.88	0.84	0.75
	ASIA	1.00	1.00	0.97	0.98	0.96
Countries	Pasca	1.00	0.98	0.95	0.82	0.60
	ASIA	1.00	1.00	1.00	1.00	0.79
Drugs	Pasca	1.00	1.00	0.96	0.92	0.75
	ASIA	1.00	1.00	1.00	1.00	0.98
Food	Pasca	0.88	0.86	0.82	0.78	0.62
	ASIA	1.00	1.00	0.93	0.95	0.90
Locations	Pasca	1.00	1.00	1.00	1.00	1.00
	ASIA	1.00	1.00	1.00	1.00	1.00
Newspapers	Pasca	0.96	0.98	0.93	0.86	0.54
	ASIA	1.00	1.00	0.98	0.99	0.85
Universities	Pasca	1.00	1.00	1.00	1.00	0.99
	ASIA	1.00	1.00	1.00	1.00	1.00
Movies	Pasca	0.92	0.90	0.88	0.84	0.79
Comedy Movies	ASIA	1.00	1.00	1.00	1.00	1.00
People	Pasca	1.00	1.00	1.00	1.00	1.00
Jazz Musicians	ASIA	1.00	1.00	1.00	0.94	0.88
Video Games	Pasca	1.00	1.00	0.99	0.98	0.98
PSP Games	ASIA	1.00	1.00	1.00	0.99	0.97
	Pasca	0.98	0.97	0.94	0.90	0.80
Average	ASIA	1.00	1.00	0.99	0.98	0.93

Table 5.5: Set instance extraction performance compared to Pasca. We report our precision for all semantic classes and at the same ranks reported in his work.

However, we should emphasize that for the last three classes: *movie*, *person*, and *video game*, ASIA did not initially produce the correct instance list given the most natural concept name. The input and (top 3) output from ASIA on those three classes are shown below:

Movies: Comedy, Action, Drama, ...
People: Musicians, Artists, Politicians, ...
Video Games: PSP, Xbox, Wii, ...

We addressed this problem by simply re-running ASIA with the first-returned class name (i.e., "Comedy Movies", "Musicians", and "PSP Games"). However, "Musicians" then returned "Jazz", "Rock", "Pop", and other kinds of music genres; thus, we re-ran ASIA with the first-returned class again (i.e., "Jazz Musicians"). The result suggests that future work is needed to support automatic construction of hypernym hierarchy using semi-structured web documents.

5.5.3 Van Durme & Paşca, 2008 and Talukdar *et al.*, 2008

Van Durme and Paşca [14] present a method for extracting large numbers of semantic classes along with their corresponding instances, based on the recombination of elements clustered through distributional similarity. The input to their algorithm is a large collection of pairs of class names and instances. They claim that this collection can be extracted using pattern-based methods such as those presented by Hearst [17]. Their experiments relied on the unstructured text available within a collection of approximately 100 million web documents in English, available in a Web repository snapshot from 2006 maintained by Google. The documents were cleaned of HTML, tokenized, split into sentences, and part-of-speech tagged using the TnT tagger [3].

Talukdar *et al.* [42] present a graph-based semi-supervised label propagation algorithm called "Adsorption" for acquiring open-domain labeled classes and their instances from a combination of unstructured and structured text sources. They construct a graph where each node represent either an instance or a class, and an edge exist between an instance node and a class node if the instance belongs to that class. The Adsorption label propagation algorithm is then applied to that graph to label all nodes based on the graph structure, ultimately producing a probability distribution over classes for each instance node. The unstructured text that they utilize was the web documents prepared and used by Van Durme & Paşca, and the structured text was the WebTables – a large database of 154 million HTML tables mined from the web [6]. Note that Adsorption requires inputs of a class name and several seed instances (five seeds per class were used in their experiments).

We compare ASIA against both Van Durme and Talukdar's systems on five classes randomly selected by Talukdar *et al.* from Van Durme's extraction outputs. Those classes are: Book Publishers, Federal Agencies, NFL Players, Scientific Journals, and Mammals. We present precision at rank 100 on the outputs from all three systems in Table 5.6 presented below:

	Precision at 100		
Class Names	**Talukdar** *et al.*, **2008**	**Van Durme & Paşca, 2008**	**ASIA**
Book Publishers	0.89	0.87	1.00
Federal Agencies	0.34	0.52	0.98
NFL Players	0.95	1.00	1.00
Scientific Journals	0.91	0.94	1.00
Mammals	0.86	1.00	0.98
Average	**0.79**	**0.87**	**0.99**

Table 5.6: Precision at rank 100 for all three systems (i.e., Talukdar *et al.*, Van Durme & Paşca, and ASIA) on five semantic classes.

Although the only input to ASIA is a class name, the results show that ASIA outperforms the other two systems, achieving an average precision of 99%, while Van Durme's system achieved 87% and Talukdar's system achieved 79%. Notice that the aims of ASIA and their work are different. ASIA focuses on extracting instances of a particular given semantic class by analyzing only documents (a subset of the web) that contain the seed words. However, in their work, they scan through every document on the web and extract instances of every semantic class identifiable by their extraction patterns.

5.5.4 Snow *et al.*, 2006

Snow [38] propose an approach that incorporates evidence from multiple classifiers over heterogenous relationships to optimize the entire structure of semantic taxonomies, using knowledge of a word's coordinate terms to help in determining its hypernyms. They apply the algorithm on the problem of sense-disambiguated noun hyponym acquisition, where they combine the predictions of hypernym and coordinate term classifiers with the knowledge in a pre-existing semantic taxonomy – WordNet 2.1. They extended the WordNet by adding 10,000 entries (synsets) at a relatively high precision of 84%.

They have made several versions of the extended WordNet available[1]. For comparison purposes, we selected the version (+30K) that achieved the best F-score (30.9%) in their experiments.

	Snow (+30k)			**Rel.**		**ASIA**		**Rel.**
Class Name	**#R.**	**#W.**	**Prec.**	**Rec.**	**#R.**	**#W.**	**Prec.**	**Rec.**
Film Directors	4	4	0.50	0.01	457	0	1.00	1.00
Manias	11	0	1.00	0.09	120	0	1.00	1.00
Canadian Provinces	10	82	0.11	1.00	10	3	0.77	1.00
Signs of the Zodiac	12	10	0.55	1.00	12	0	1.00	1.00
Roman Emperors	44	4	0.92	0.47	90	0	1.00	0.96
Academic Departments	20	0	1.00	0.67	27	0	1.00	0.90
Choreographers	23	10	0.70	0.14	156	0	1.00	0.94
Elected Officials	5	102	0.05	0.31	12	0	1.00	0.75
Double Stars	11	1	0.92	0.46	20	0	1.00	0.83
South American Countries	12	1	0.92	1.00	12	0	1.00	1.00
Prizefighters	16	4	0.80	0.23	63	1	0.98	0.89
Newspapers	20	0	1.00	0.23	71	0	1.00	0.81
Average	15.7	18.2	0.70	0.47	87.5	0.3	0.98	0.92

Table 5.7: A comparison of number of (R)ight and (W)rong instances, precision, and relative recall between Snow's Wordnet (+30k) and ASIA.

For the experimental comparison, we focused on leaf semantic classes from the extended WordNet that have many hyponyms, so that a meaningful comparison could be made: specifically, we selected nouns that have at least three hyponyms, such that the hyponyms are the leaf nodes in the hypernym hierarchy of WordNet. Of these, 210 were extended by Snow. Preliminary experiments showed that (as in the experiments with Pasca's classes above) ASIA did not always extracted instances of the intended meaning; to avoid this problem, we instituted a second filter, and discarded ASIA's results if the intersection of hyponyms from ASIA and WordNet constituted less than half of those in WordNet. About 50 of the 210 nouns passed this filter. Finally, we manually evaluated precision and recall of a randomly selected set of twelve of these 50 nouns.

We present the results in Table 5.7. We use a cut-off threshold of 0.26 based on the threshold trained by 5-fold cross validations on TREC 13-15 presented in Table 3.5 in

[1] `http://ai.stanford.edu/~rion/swn/`

order to truncate the ranked list produced by ASIA, so that we can compute precision. Since only a few of these twelve nouns are closed sets, we cannot generally compute recall; instead, we define *relative recall* to be the ratio of correct instances to the union of correct instances from both systems. As shown in the results, ASIA has much higher precision (98% vs. 70%), and much higher relative recall (92% vs. 47%). When we evaluated Snow's extended WordNet, we assumed all instances that were in the original WordNet are correct (even though some are not). The three incorrect Canadian provinces from ASIA are actually the three Canadian territories, and the one incorrect prizefighter is "World Boxing Council".

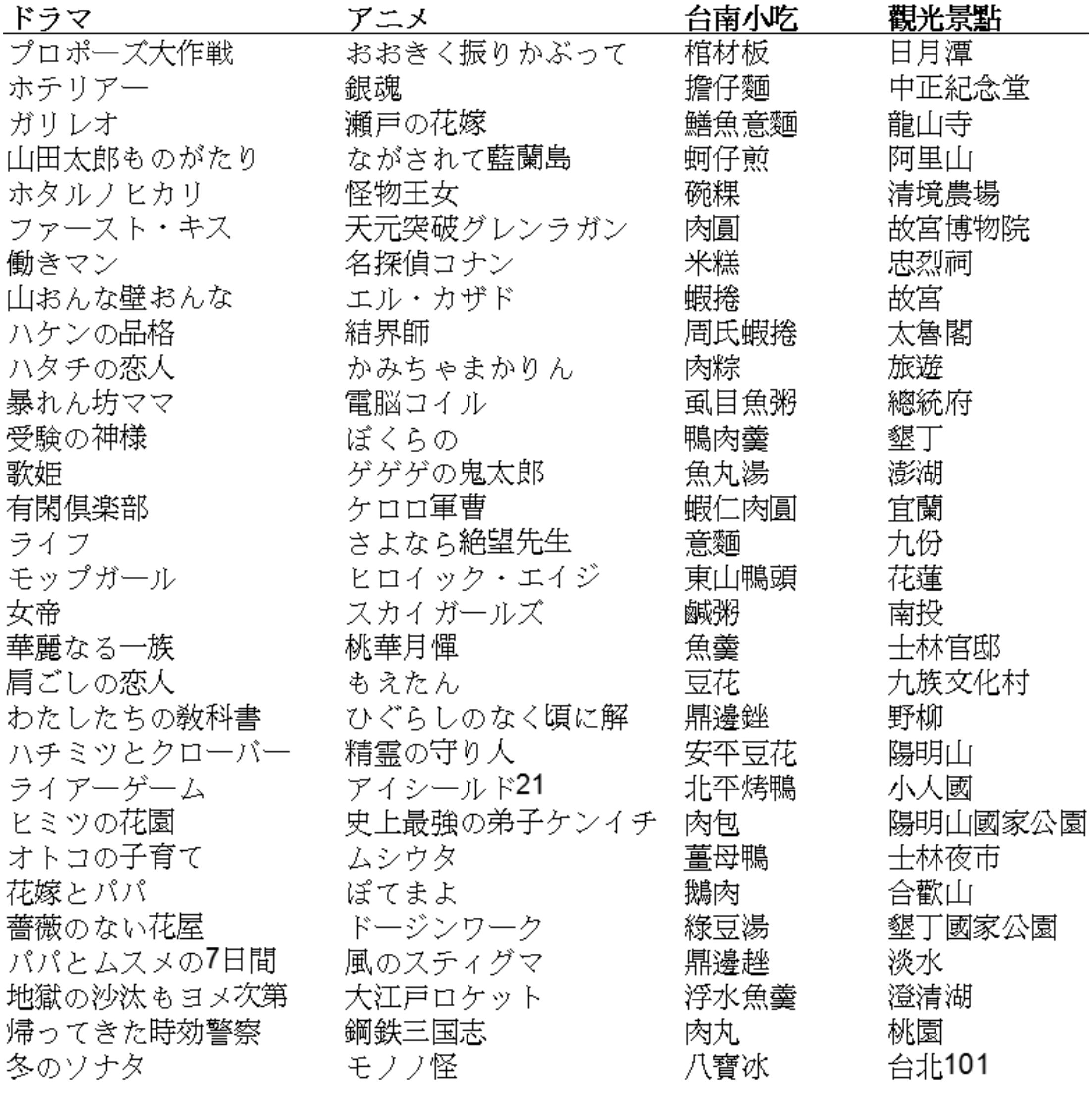
5.6 Summary

In this chapter, we have shown that ASIA, a SEAL-based system, extracts set instances with high precision and recall in multiple languages given only the set name. It obtains a high MAP score (93%) averaged over 36 benchmark problems in three languages (Chinese, Japanese, and English). ASIA's approach is based on web-based set expansion using semi-structured documents, and is motivated by the conjecture that for many natural classes, the amount of information available in semi-structured documents on the Web is much larger than the amount of information available in free-text documents. This conjecture is given some support by our experiments: for instance, ASIA finds 457 instances of the set "film director" with perfect precision, whereas Snow *et al*'s state-of-the-art methods for extraction from free text extract only four correct instances, with only 50% precision.

ASIA's approach is also quite language-independent. By adding a few simple hyponym patterns, we can easily extend the system to support other languages. We have also shown that Hearst's method works not only for English, but also for other languages such as Chinese and Japanese. We note that the ability to construct semantic lexicons in diverse languages has obvious applications in machine translation. We have also illustrated that ASIA outperforms five other English systems [14, 20, 28, 38, 42], even though many of these use more input than just a semantic class name. In addition, ASIA is also quite efficient, requiring only a few minutes of computation and couple hundreds of web pages per problem. The ability to construct semantic lexicons in diverse languages has obvious applications in machine translation. In the future, we plan to investigate the possibility of constructing hypernym hierarchy automatically using semi-structured documents. We also plan to explore whether lexicons can be constructed using only the *back-off* method for hyponym extraction, to make ASIA completely language independent. We also wish to explore whether performance can be improved by simultaneously finding class instances in multiple languages (e.g., Chinese and English) while learning translations between the extracted instances.

Time Travel Movies	**Simpsons Characters**	**Nobel Prize Winners**
the time machine	marge simpson	nelson mandela
back to the future	krusty the clown	woodrow wilson
time bandits	bart simpson	albert schweitzer
somewhere in time	lisa simpson	mother teresa
peggy sue got married	homer simpson	elie wiesel
time after time	ralph wiggum	mikhail gorbachev
millennium	ned flanders	jimmy carter
frequency	maggie simpson	desmond tutu
the final countdown	comic book guy	linus pauling
timeline	waylon smithers	theodore roosevelt
donnie darko	kent brockman	willy brandt
the terminator	apu nahasapeemapetilon	aung san suu kyi
the philadelphia experiment	carl carlson	amnesty international
terminator	chief wiggum	elihu root
planet of the apes	milhouse van houten	yitzhak rabin
lost in space	nelson muntz	menachem begin
groundhog day	groundskeeper willie	henry kissinger
kate and leopold	barney gumble	shimon peres
flight of the navigator	montgomery burns	cordell hull
timecop	sideshow bob	shirin ebadi
butterfly effect	lenny leonard	frank b. kellogg
happy accidents	hans moleman	andrei sakharov
army of darkness	moe szyslak	fridtjof nansen
time machine	martin prince	arthur henderson
star trek iv: the voyage home	edna krabappel	wangari maathai
black knight	lionel hutz	norman borlaug
freejack	abraham simpson	kim dae jung
apes	troy mcclure	carl von ossietzky
deja vu	professor frink	henri la fontaine
terminator 2: judgment day	seymour skinner	yasser arafat

Table 5.8: Real example inputs and outputs from ASIA in English.



ドラマ	アニメ	台南小吃	觀光景點
プロポーズ大作戦	おおきく振りかぶって	棺材板	日月潭
ホテリアー	銀魂	擔仔麵	中正紀念堂
ガリレオ	瀬戸の花嫁	鱔魚意麵	龍山寺
山田太郎ものがたり	ながされて藍蘭島	蚵仔煎	阿里山
ホタルノヒカリ	怪物王女	碗粿	清境農場
ファースト・キス	天元突破グレンラガン	肉圓	故宮博物院
働きマン	名探偵コナン	米糕	忠烈祠
山おんな壁おんな	エル・カザド	蝦捲	故宮
ハケンの品格	結界師	周氏蝦捲	太魯閣
ハタチの恋人	かみちゃまかりん	肉粽	旅遊
暴れん坊ママ	電脳コイル	虱目魚粥	總統府
受験の神様	ぼくらの	鴨肉羹	墾丁
歌姫	ゲゲゲの鬼太郎	魚丸湯	澎湖
有閑倶楽部	ケロロ軍曹	蝦仁肉圓	宜蘭
ライフ	さよなら絶望先生	意麵	九份
モップガール	ヒロイック・エイジ	東山鴨頭	花蓮
女帝	スカイガールズ	鹹粥	南投
華麗なる一族	桃華月憚	魚羹	士林官邸
肩ごしの恋人	もえたん	豆花	九族文化村
わたしたちの教科書	ひぐらしのなく頃に解	鼎邊銼	野柳
ハチミツとクローバー	精霊の守り人	安平豆花	陽明山
ライアーゲーム	アイシールド21	北平烤鴨	小人國
ヒミツの花園	史上最強の弟子ケンイチ	肉包	陽明山國家公園
オトコの子育て	ムシウタ	薑母鴨	士林夜市
花嫁とパパ	ぽてまよ	鵝肉	合歡山
薔薇のない花屋	ドージンワーク	綠豆湯	墾丁國家公園
パパとムスメの7日間	風のスティグマ	鼎邊趖	淡水
地獄の沙汰もヨメ次第	大江戸ロケット	浮水魚羹	澄清湖
帰ってきた時効警察	鋼鉄三国志	肉丸	桃園
冬のソナタ	モノノ怪	八寶冰	台北101

Figure 5.6: Real example inputs and outputs of ASIA in Japanese and Chinese. From left to right, the semantic class names are “Dramas”, “Animations”, “Tainan Small Eats”, and “Attractions”.

Chapter 6

Bilingual SEAL

6.1 Introduction

Bootstrapping is an iterative process for a system to continuously improve its own performance by utilizing its own outputs. As illustrated by our Iterative SEAL, or iSEAL, in Chapter 4, bootstrapping may perform poorly if incorrect candidate instances are chosen and used as seeds. In this chapter, we propose an approach that utilizes redundant information to minimize the chance of choosing incorrect candidate instances, and thus improving the overall performance. We introduce an extended version of SEAL called Bilingual SEAL, which expands two sets of instances alternately by using two separate instances of iSEAL. Both sets represent the same semantic class but each in a different human language (e.g., Disney movies in English and Chinese). To verify the correctness of a candidate instance in one language (e.g., English), we translate the instance to another language (e.g., Chinese) and ensure that its translation also exists in the expanded set of that language (e.g., Chinese).

In this chapter, we describe our proposed approach in Section 6.2 and our translation method in Section 6.3. In Section 6.4, we explain our experiments and present our experimental results. We summarize this chapter in Section 6.5.

6.2 Proposed Approach

We configure iSEAL to bootstrap with increasing seed size (ISS) using Random Walk [43], described in Section 2.4.3, because this configuration has been shown experimentally to be most robust to noisy seeds as illustrated in Figure 4.4. Our proposed

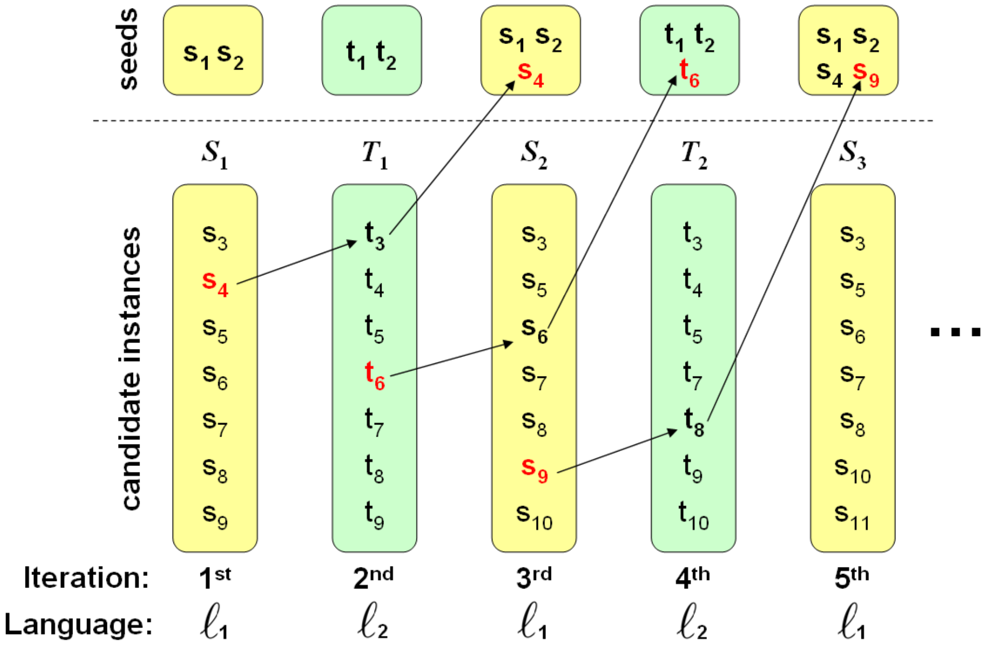
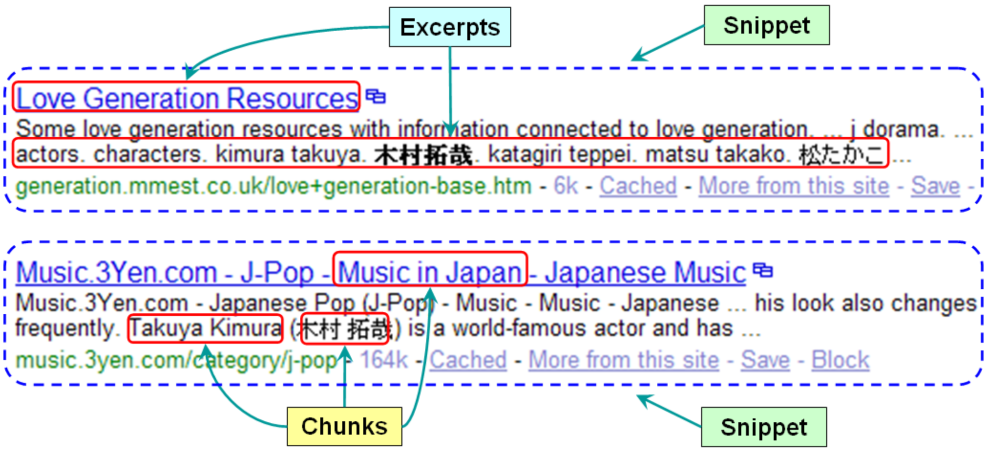

seeds
s_1 s_2
t_1 t_2
s_1 s_2 s_4
t_1 t_2 t_6
s_1 s_2 s_4 s_9
candidate instances
S_1
T_1
S_2
T_2
S_3
s_3 s_4 s_5 s_6 s_7 s_8 s_9
t_3 t_4 t_5 t_6 t_7 t_8 t_9
s_3 s_5 s_6 s_7 s_8 s_9 s_{10}
t_3 t_4 t_5 t_7 t_8 t_9 t_{10}
s_3 s_5 s_6 s_7 s_8 s_{10} s_{11}
...
Iteration: 1st 2nd 3rd 4th 5th
Language: ℓ_1 ℓ_2 ℓ_1 ℓ_2 ℓ_1

Figure 6.1: An illustration of our proposed approach.

approach requires an input of two seeds in one language ℓ_1 and two seeds in another language ℓ_2 where both pairs of seeds are instances of the same semantic class (e.g., Disney movies). We bootstrap the two input pairs of seeds separately and alternately; thus, the first iteration expands seeds in ℓ_1, the second expands seeds in ℓ_2, third expands seeds in ℓ_1, and so on. At every iteration, in order to choose a candidate instance to be used as the new seed for the current i^{th} iteration, we choose an instance from $(i-2)^{th}$ iteration, whose translation exist in $(i-1)^{th}$ iteration, by utilizing a named entity translator which is described in the next section. The basic idea is that we want to ensure that the instance we choose as the new seed exists in one expanded set while its translation exists in another. Note that each expansion for a language extends the graph previously constructed for that language only; so there will only be two graphs that expand alternately. At the end, there will be two expanded sets of instances, each in a different language.

Figure 6.1 illustrates an example of our approach. Suppose a user issues a two-seed query, s_1 and s_2, in one language as well as a two-seed query, t_1 and t_2, in another language, we first expand s_1 and s_2 to obtain a ranked list S_1 and also expand t_1 and t_2 to obtain a ranked list T_1. As we traverse down list S_1, we translate every encountered



Love Generation Resources

Some love generation resources with information connected to love generation. ... j dorama. ... actors. characters. kimura takuya. 木村拓哉. katagiri teppei. matsu takako. 松たかこ ...

generation.mmest.co.uk/love+generation-base.htm - 6k - Cached - More from this site - Save -

Music.3Yen.com - J-Pop - Music in Japan - Japanese Music

Music.3Yen.com - Japanese Pop (J-Pop) - Music - Music - Japanese ... his look also changes frequently. Takuya Kimura (木村 拓哉) is a world-famous actor and has ...

music.3yen.com/category/j-pop - 164k - Cached - More from this site - Save - Block

Figure 6.2: Examples of snippets, excerpts, and chunks.

instance until we find an instance whose top-three translations exist in top-half of T_1. In the example, s_4 is found to have a translation in T_1 (i.e., t_3); so it is used as the new seed for the third expansion to obtain S_2. Now, this time we traverse down T_1 to find an instance whose top-three translations exist in top-half of S_2. In the example, t_6 is found to have a translation in S_2 (i.e., s_6); so it is used as the new seed for the fourth expansion to obtain T_2, and so on. These procedures are repeated until no new instances emerge or until a fixed number of expansions has been reached.

6.3 Named Entity Translation

It is not trivial to automatically translate named entities since new names emerge all the time. In this section, we introduce a component in the Bilingual SEAL called Automatic Named Entity Translator (ANET), which automatically discovers the translation of any given name using the Web. This is achieved by identifying words in the target language that frequently and closely co-occur with the words in the source language from bilingual search results returned by Yahoo! search engine. ANET's approach is language-independent because it does not require any language-specific transliteration models. However, its performance would suffer substantially if the character sets of the source and target language are not disjoint (e.g., Chinese and Japanese) because it cannot easily distinguish the language of a candidate name.

Below, we refer to each of the search results as a *snippet*, each of the sentences

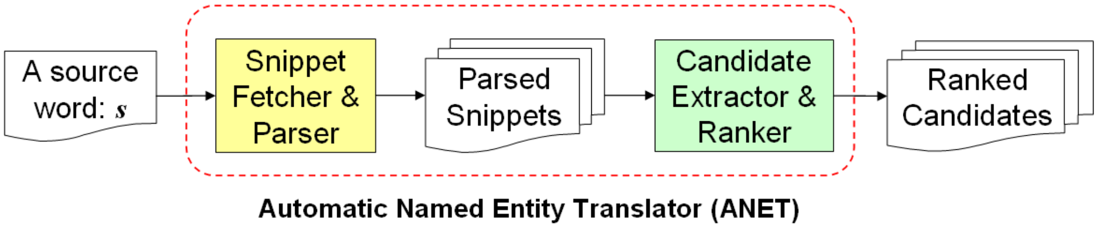


Figure 6.3: Flow chart of the ANET system.

delimited by "..." in a snippet as an *excerpt*, and each sequence of characters bounded by punctuation marks or foreign-language words as a *chunk*. These notions has been introduced in Section 5.3.1 and illustrated by Figure 5.3, but for convenience, we include the same figure here as Figure 6.2. In addition, we denote a name in the source language as s and the translation of the name in the target language as t.

Figure 6.3 illustrates the architecture of ANET, which follows the procedures described below:

1. Send s as a query to Yahoo! and request documents written in the target language.
2. Retrieve and parse the top one-hundred returned bilingual snippets.
3. Extract all chunks in the target language (to be used as translation candidates) from the snippets.
4. Compute scores for and rank those candidates.

In the first step above, ANET attempts to retrieve bilingual snippets containing both s and t, by sending s as a search query to Yahoo! along with a parameter specifying Yahoo! to return only those web pages written in the target language. In the second step, ANET parses each snippet into a structured format containing elements such as the title, page URL, and list of excerpts. In the third step, ANET extracts a set of translation candidates of s. A translation candidate of s is defined as a chunk in the target language which co-occurs with s in any excerpt. Consider the example shown in Figure 6.2 where the target language is English and the source word s is a famous actor's name in Japanese:

木村拓哉

since the chunk "Takuya Kimura" occurs in the same excerpt as s, it is a potential translation candidate of s (in fact, it is a correct translation). In the fourth step, ANET ranks translation candidates by assigning each candidate c a weight determined by using a function almost identical to the $weight(c)$ presented in Section 5.3.1, which is a heuristic score based on how frequently c co-occurs with s and how closely c occurs with s in the excerpts. The only difference is that the *hyponym pattern frequency* is removed from the equation. More specifically, the scoring function is defined as:

$$weight(c) = \frac{sf(c, \mathbf{S})}{|\mathbf{S}|} \times \frac{ef(c, \mathbf{E})}{|\mathbf{E}|} \times \frac{wcf(c, \mathbf{E})}{|\mathbf{C}|}$$

where $\mathbf{S}$ the set of snippets, $\mathbf{E}$ the set of excerpts, and $\mathbf{C}$ the set of chunks. The function $sf(c, \mathbf{S})$ is the snippet frequency of c (i.e., the number of snippets containing c), and $ef(c, \mathbf{E})$ is the excerpt frequency of c (i.e., the number of excerpts containing c). Furthermore, $wcf(c, \mathbf{E})$ is the weighted chunk frequency of c, which is defined as follows:

$$wcf(c, \mathbf{E}) = \sum_{e \in \mathbf{E}} \sum_{c \in e} \frac{1}{dist(c, e) + 1}$$

where $dist(c, e)$ is the number of characters between the candidate c and the source word s in the excerpt e. This model weights every occurrence of c based on the assumption that chunks closer to a source word are usually more important than those further away. It also heavily rewards frequency since chunks that co-occur frequently with the source word are more important than those that co-occur infrequently.

6.4 Experiments

6.4.1 Experimental Setting

We evaluated Bilingual SEAL by conducting two experiments. The first experiment was in Chinese and English, and the second in Japanese and English. The reason that we did not conduct a third experiment in Chinese and Japanese is because, as mentioned earlier, ANET's performance suffers substantially when the character sets of the source and target language are not disjoint; it cannot easily distinguish the language of a candidate name.

For each experiment, we created two instances of iSEAL, one for each language. We initiated the bootstrapping process by providing each iSEAL with two supervised seeds, and we configure iSEAL to bootstrap with increasing seed size using random

	# of Unique Translations	**Baseline Precision**	**By-Product Precision**	**Δ**
Eng to Chi	89	0.865	0.910	5.2%
Chi to Eng	85	0.635	0.788	24.1%
Eng to Jap	83	0.892	0.988	10.8%
Jap to Eng	88	0.477	0.886	85.7%
Average	**86**	**0.717**	**0.893**	**24.5%**

Table 6.1: Precisions of translation pairs generated 1) as **by-products** from the bilingual bootstrapping, and 2) by directly translating the source words in the by-product dictionaries using ANET to serve as **baselines**.

walk as described in Section 4.2.2.2. To make the problem more challenging and accentuate differences between approaches, at every iteration, iSEAL retrieved only the top ten web pages from Yahoo! and accumulated statistics by growing the graph required by the random-walk ranking scheme. The evaluation metric used is the Mean Average Precision (MAP), which is described in Section 2.5.5. As for the evaluation datasets, since Bilingual SEAL expands instances of the same semantic class in multiple languages, we used only the first nine semantic classes listed in Table 2.7 because they have datasets in multiple languages. In each experiment, we conducted ten bootstrapping iterations (five per each language) independently three times for each class, and present their average performance. As for baselines, we present results of monolingual bootstrapping (i.e., without using ANET) for each language in each experiment.

6.4.2 Experimental Results

Figure 6.4 illustrates the results of the first experiment in Chinese and English, with details shown in Table 6.2. Figure 6.5 presents the results of the second experiment in Japanese and English, with details shown in Table 6.3. The error bars in those figures are the standard errors of the mean, and to simplify the graph we only show error bars for the best-performing method (standard errors for the other methods are comparable). As illustrated, both experiments show that bilingual bootstrapping performs better than monolingual bootstrapping. More specifically, in the first experiment, the Chinese final results using bilingual bootstrapping are 11% better than using monolingual bootstrapping (93.8% vs. 84.5%), and its corresponding English final results are

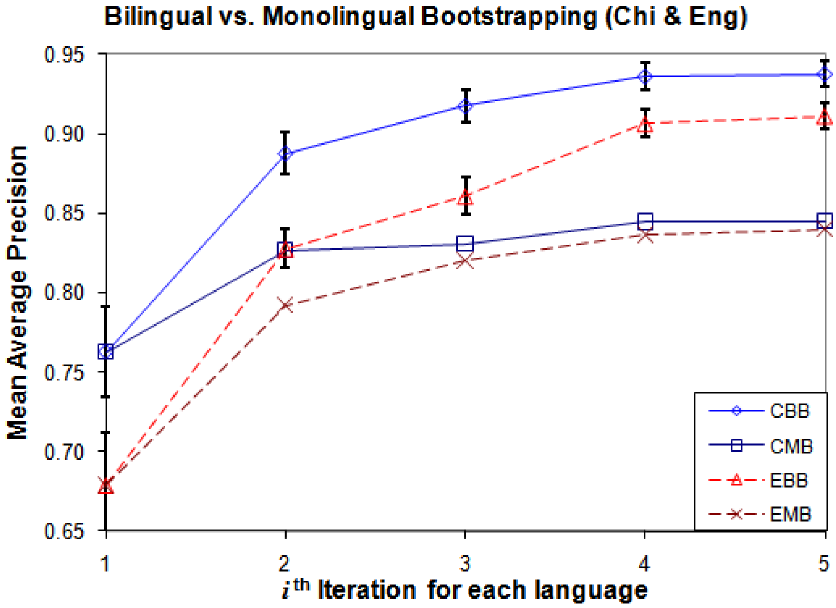


Figure 6.4: Performance of bilingual and monolingual bootstrapping in Chinese and English. CBB and CMB are the Chinese results in bilingual and monolingual bootstrapping respectively. EBB and EMB are the English results in bilingual and monolingual bootstrapping respectively.

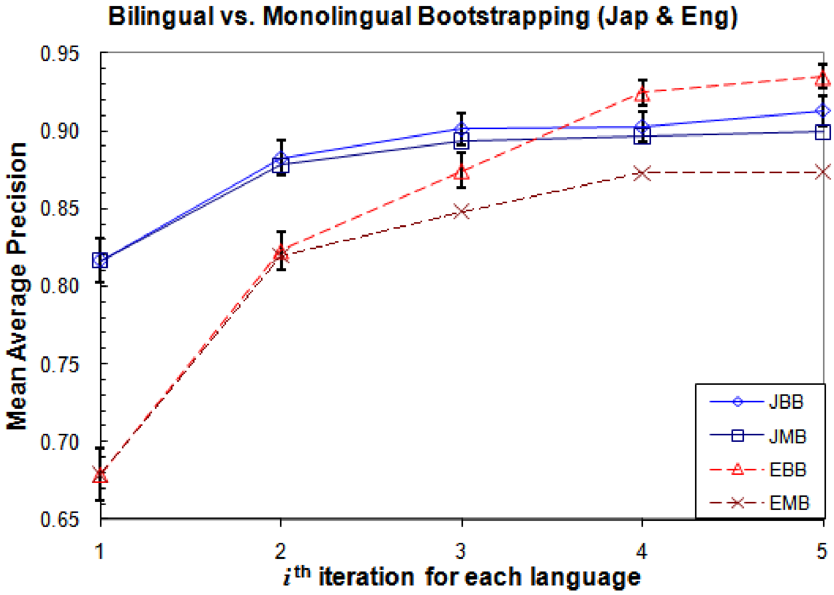


Figure 6.5: Performance of bilingual and monolingual bootstrapping in Japanese and English. JBB and JMB are the Japanese results in bilingual and monolingual bootstrapping respectively.

	Chi.	**Mono**	**Bi**		**Eng.**	**Mono**	**Bi**	
	1st	**5th**	**5th**	**Δ**	**1st**	**5th**	**5th**	**Δ**
disney-movies	0.641	0.644	0.878	36.4%	0.482	0.576	0.773	34.1%
constellations	0.948	1.000	1.000	0.0%	0.867	0.868	0.864	-0.4%
countries	0.672	0.694	0.830	19.5%	0.598	0.673	0.859	27.6%
mlb-teams	1.000	1.000	1.000	0.0%	0.946	1.000	1.000	0.0%
nba-teams	0.818	0.986	0.999	1.3%	0.328	0.995	0.999	0.4%
nfl-teams	0.848	0.991	0.993	0.2%	1.000	1.000	1.000	0.0%
car-makers	0.202	0.493	0.857	73.6%	0.453	0.455	0.713	56.8%
us-presidents	0.791	0.809	0.888	9.7%	0.729	0.995	0.995	0.0%
us-states	0.945	0.987	0.995	0.8%	0.707	1.000	1.000	0.0%
Average	**0.763**	**0.845**	**0.938**	**11.0%**	**0.679**	**0.840**	**0.911**	**8.5%**

Table 6.2: Performance (MAP) of monolingual and bilingual bootstrapping in Chinese and English at 1^{st} and 5^{th} iteration for each language, as well as the relative improvement of bilingual bootstrapping over monolingual bootstrapping at the 5^{th} iteration.

	Jap.	**Mono**	**Bi**		**Eng.**	**Mono**	**Bi**	
	1st	**5th**	**5th**	**Δ**	**1st**	**5th**	**5th**	**Δ**
disney-movies	0.523	0.697	0.753	8.2%	0.482	0.666	0.788	18.4%
constellations	0.977	1.000	1.000	0.0%	0.867	0.854	0.992	16.1%
countries	0.616	0.655	0.657	0.4%	0.598	0.835	0.927	11.1%
mlb-teams	0.944	0.999	0.996	-0.3%	0.946	1.000	1.000	0.0%
nba-teams	1.000	1.000	1.000	0.0%	0.328	1.000	1.000	0.0%
nfl-teams	0.971	0.995	0.995	0.0%	1.000	1.000	1.000	0.0%
car-makers	0.460	0.857	0.882	2.9%	0.453	0.512	0.707	38.2%
us-presidents	0.857	0.891	0.936	5.0%	0.729	0.995	1.000	0.5%
us-states	1.000	1.000	1.000	0.0%	0.707	1.000	1.000	0.0%
Average	**0.817**	**0.899**	**0.913**	**1.6%**	**0.679**	**0.873**	**0.935**	**7.0%**

Table 6.3: Performance (MAP) of monolingual and bilingual bootstrapping in Japanese and English at 1^{st} and 5^{th} iteration for each language, as well as the relative improvement of bilingual bootstrapping over monolingual bootstrapping at the 5^{th} iteration.

9% better (91.1% vs. 84.0%). In the second experiment, the Japanese final results using bilingual bootstrapping are 2% better (91.3% vs. 89.9%) and its corresponding English final results are 7% better (93.5% vs. 87.3%). The results show that set expansion performs better by using two languages than using only one language.

We collected the translation pairs (i.e., pairs of source and target words) that are produced, as *by-products*, from the bilingual bootstrapping. These by-products could be used as bilingual dictionaries for numerous purposes (e.g., machine translation). For each experiment, we evaluated the precision of the by-product dictionaries, and present the results in Table 6.1. For each experiment, we also evaluated the precisions of *baseline* dictionaries generated by directly translating the source words in the by-product dictionaries using ANET (i.e., without bilingual bootstrapping). The results show that the precision of the by-product dictionaries have an average of 25% improvement over the baseline dictionaries (72% vs. 89%). The results illustrate that named entity translation also performs better by using two languages.

6.5 Summary

In this chapter, we have shown that the performance of set expansion can be improved by exploiting redundant information of classes in different languages. More specifically, we illustrated that the performance of set expansion in Chinese and English can be improved dramatically by coupling them together. This is achieved by bootstrapping instances in these two languages alternately, of which the instances in different languages are bridged together by a simple named entity translator. Same results were observed by coupling Japanese and English. In addition, we have also shown that the precision of translation pairs generated as by-products from our bilingual-bootstrapping approach has also improved dramatically for both Chinese-English and Japanese-English pairs.

Chapter 7

Relational SEAL

7.1 Introduction

SEAL was initially designed to handle only unary relationships (e.g., "x is a CEO"). In this chapter, we show that SEAL's character-level analysis techniques can be readily extended to handle binary relationships (e.g., "x is the CEO of company y"). We introduce a system called Relational SEAL, which is an extension of iSEAL to learn binary relational concepts from a small number of seeds. An example of input instances to our (binary) relational set expansion system is "*Bill Gates / Microsoft*" and "*Larry Page / Google*", of which our system produces other instance pairs of the same relation such as "*Larry Ellison / Oracle*".

In this chapter, we explore the impact on relational set-expansion performance of one of the innovations in SEAL, specifically, the use of character-level techniques to detect wrappers in semi-structured web pages. We show that, as with unary relationships, MAP performance is 26 points lower when wrappers are restricted to be HTML-related. We also show that the Relational SEAL has good performance on our binary evaluation datasets, which includes English and Chinese, thus demonstrating language-independence. Furthermore, we illustrate that the learning of binary concepts can also be bootstrapped by the strategy described earlier in Section 4.2.2.2 to improve its performance.

Although some early systems for web-page analysis induce rules at character-level, such as WIEN [21] and DIPRE [4], most recent approaches for extracting relations have been conducted on free text requiring some language-dependent parsers or taggers for

English [1, 7, 29, 31, 38, 41]. Much research has also been done on extracting relations from HTML-structured table [6, 15, 25, 44]; however, they all incorporated heuristics by using HTML parsers for exploiting HTML structures; thus, they cannot handle documents written in other mark-up languages.

In this chapter, Section 7.2 describes the method for extending SEAL to handle binary relationships, Section 7.3 presents experimental results, and we compare our results to a prior work in Section 7.4. Lastly, we summarize this chapter briefly in Section 7.5.

7.2 Identifying Wrappers for Binary Relations

We extend the wrapper construction algorithm described in Section 2.3.2 to support relational set expansion. The major difference is that we introduce a third type of context called the *middle* context that occurs between the left and right contexts of a wrapper for separating any two instances. We execute the wrapper construction algorithm described in Table 2.2, except that a seed instance in the algorithm is now a seed instance pair bracketing some middle context (i.e., "$s_1 \cdot$ *middle*$\cdot$ s_2").

Given some seed pairs (i.e., s_1 and s_2), the algorithm first locates the following two strings in some given documents: "s_1`[...]`s_2" and its reversed version: "s_2`[...]`s_1", where "`[...]`" is a wildcard matching any sequence of characters not containing any of the two seeds. For every pair of seeds located, SEAL extracts their left, middle (i.e., "`[...]`"), and right contexts. The left and right contexts are inserted into their corresponding tries, while the middle context is inserted into a list. Every middle context is assigned a flag indicating whether the two instances bracketing it were found in the same or reversed order as the input seed pairs. Every entry in the seed instance list now stores a pair of instances as one single string (i.e., "s_1/s_2"). An *id* stored in a node now matches the index of an instance pair as well as a middle context. Shown below is a mock example document of which all instances in the following three seed pairs are located (and underlined): "*Ford / USA*", "*Nissan / Japan*", "*Toyota / Japan*".

```
GtpKxHNissanoKpJapanxjHJgleTuoLpBlcLBxKH
FordEFcUSAxkrpWNapnIkAAHOFordawHDaUSAuoh
deQsKxHAcuraoKpJapanxjHdIjWnOxoToyotaVaq
JapanzxKHAudiEFcGermanyxkrOyQKxHToyotaoK
pJapanxjHCRdmtqOVKxHFordoKpUSAxjHaJASzEi
nSfrlaFordLMmpUSAofwNLWxKHToyotaEFcJapan
xkrHxQKzrHpoKdGEKtxKHNissanEFcJapanxkrEq
```

After performing the abovementioned procedures on this mock document, we now have context tries that are much more complicated than those illustrated in Figure 2.3. We also have a list of middle contexts of which a sample of it is shown below:

id	Seed Pairs	*r*	Middle Context
0	Nissan/Japan	No	`oKp`
1	Nissan/Japan	No	`EFc`
2	Nissan/Japan	Yes	`xkrHxQKzrHpoKd...`
4	Toyota/Japan	No	`oKp`
6	Toyota/Japan	Yes	`xjHdIjWnOxo`
9	Ford/USA	No	`EFc`
13	Ford/USA	Yes	`xkrpWNapnIkAAHO`

where r indicates if the two instances bracketing the middle context were found in the reversed order as the input seed pairs. In order to find the maximally long contextual strings, the "Intersect" function in the wrapper construction pseudo-code presented in Table 2.2 needs to be replaced with the following:

```
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 same middle context
```

which returns those seed pairs that 1) are bracketed by the strings associated with the two input nodes and 2) have the same middle context. A wrapper for relational set expansion, or *relational wrapper*, is defined by the left, middle, and right contextual strings. The relational wrappers constructed from the mock document given the example seed pairs are shown below. Notice that two additional instance pairs are discovered: "*Audi / Germany*" and "*Acura / Japan*".

Wrapper:	xKH[.1.]EFc[.2.]xkr
Content:	**audi/germany**, ford/usa, nissan/japan, toyota/japan
Wrapper:	KxH[.1.]oKp[.2.]xjH
Content:	**acura/japan**, ford/usa, nissan/japan, toyota/japan

Below, we show real examples of wrappers constructed and candidate instances extracted from some randomly-sampled web pages given two instance pairs of car makers and their headquartered countries: "*Mazda / Japan*" and "*Venturi / France*". More examples are presented at the end of this chapter.

Seeds:	**Mazda / Japan, Venturi / France**
URL:	http://www.jrfilters.com/filtres/index.php?lng=en
Wrapper:	&page=filtres&lng=en">[.1.] ([.2.])</option><option value="index.php?
URL:	http://www.jrfilters.com/suspensions/index.php?famille=1&lng=en
Wrapper:	&lng=en">[.1.] ([.2.])</option><option value="index.php?famille=1&rubrique1
URL:	http://www.street-car.net/forums/forumdisplay.php?f=10
Wrapper:	"><strong>[.1.]</strong></a> </div> <div class="smallfont">Country of origin:[.2.].
URL:	http://www.allcarcentral.com/
Wrapper:	file.html">[.1.],[.2.]</a><br />
Content:	abarth / italy, acura / japan, alfa romeo / italy, aston martin / england, auburn / usa, audi / germany, austin healey / england, austin / england, auto union / germany, balwin / usa, bandini / italy, bentley / england, bmw / germany, brabham / england, bricklin / usa, bristol / england, brm / england...

7.3 Experiments

For binary relations, we performed the same experiment as with unary relations described in Section 2.5. Again, we investigated whether HTML-based relational wrappers are more effective than character-based relational wrappers.

7.3.1 Experimental Setting

In Table 7.1, we introduce four types of HTML-based relational wrappers, referred to as R1 to R4, which are extended from the HTML-based unary wrappers presented in

	Left`[...]`	**Middle**	`[...]`**Right**
RE	`.+[...]`	`.+`	`[...].+`
R1	`.*[<>].*[...]`	`.*[<>].*`	`[...].*[<>].*`
R2	`.*>[...]`	`<.*>`	`[...]<.*`
R3	`.*<.+?>.*[...]`	`.*<.+?>.*`	`[...].*<.+?>.*`
R4	`.*<.+?>[...]`	`<.+>`	`[...]<.+?>.*`

Table 7.1: Regular expressions illustrating character restrictions for our proposed relational extractor (RE) and the four types of HTML-based relational wrappers (R1-R4).

Dataset	**Instance**$_1$ vs.	**Instance**$_2$	**Lan**$_1$	**Lan**$_2$	**Size**
us-governors	US State/Territory vs.	Governor	Eng	Eng	56
taiwan-mayors	Taiwanese City vs.	Mayor	Chi	Chi	26
nba-teams	NBA Team (Chi) vs.	NBA Team (Eng)	Chi	Eng	30
fed-agencies	Fed. Agency Acronym vs.	Full Name	Eng	Eng	387
car-makers	Car Manufacturer vs.	Headquartered Country	Eng	Eng	122

Table 7.2: The description and language of each instance in the five relational datasets for evaluating Relational SEAL.

Table 2.6. These relational wrappers possess the same properties as the unary wrappers, that is, they have increasing constraints from type 1 (less strict) to 4 (more strict). In this section, we refer to our relational extractor proposed in Section 7.2 as RE.

In order to evaluate our approach, we manually constructed five relational datasets, as described in Table 7.2. For the last two datasets, since there are too many instances, we tried our best to make the lists as exhaustive as possible. Please refer to Appendix B for the content of the five relational datasets.

To evaluate relational wrappers, we performed relational set expansion on randomly selected seeds from the five relational datasets. For every dataset, we select two seeds randomly and bootstrap the relational set expansion ten times, using the increasing seed size (ISS) strategy described in Section 4.2.2.2. We repeat this step three times for every dataset and present the average performance.

7.3.2 Experimental Results

The results after the first iteration are shown in Table 7.3 and after the tenth iteration in Table 7.4. Before computing precision at 100 for each resulting list, we remove all

	Mean Avg. Precision					**Precision@100**				
Dataset	**RE**	**R1**	**R2**	**R3**	**R4**	**RE**	**R1**	**R2**	**R3**	**R4**
us-governors	97.4	89.3	89.2	89.3	89.2	55	50	51	50	50
taiwan-mayors	99.8	95.6	94.3	91.3	90.8	25	25	24	23	23
nba-teams	100.0	99.9	99.9	99.9	99.2	30	30	30	30	30
fed-agencies	43.7	14.5	5.2	11.1	5.2	96	55	20	40	20
car-makers	61.7	0.0	0.0	0.0	0.0	74	0	0	0	0
Average	80.5	59.9	57.7	58.3	56.9	56	32	25	29	25

Table 7.3: Performance of our proposed relational extractor (RE) and various types of HTML-based relational wrappers (R1-R4) on the five relational datasets after **first** iteration.

	Mean Avg. Precision					**Precision@100**				
Dataset	**RE**	**R1**	**R2**	**R3**	**R4**	**RE**	**R1**	**R2**	**R3**	**R4**
us-governors	98.9	97.0	95.3	94.1	93.9	55	55	54	53	53
taiwan-mayors	99.8	98.3	96.9	93.8	94.3	25	25	25	24	24
nba-teams	100.0	100.0	99.2	98.4	98.6	30	30	30	30	30
fed-agencies	65.5	54.5	27.9	55.3	30.0	97	97	61	95	69
car-makers	81.6	0.0	0.0	0.0	0.0	90	0	0	0	0
Average	89.2	70.0	63.9	68.3	63.4	59	41	34	40	35

Table 7.4: Performance of our proposed relational extractor (RE) and various types of HTML-based relational wrappers (R1-R4) on the five relational datasets after **tenth** iteration.

synonyms for every instance from the list and keep only the top-most-ranked synonym; this ensures that every instance is unique in the list. Notice that for the "car-makers" dataset, there exist no HTML-based wrappers, thus resulting in zero performance for R1 to R4. In each table, the results indicate that character-based wrappers perform the best, while those HTML-based wrappers that require tight HTML bracketing of instances (i.e., R2 and R4) perform the worse.

In addition, the results illustrate that bootstrapping is effective for expanding relational instance pairs. As illustrated in Table 7.4, the result of finding translation pairs of NBA team names is perfect, and it is almost perfect for finding pairs of U.S. states/territories and governors, as well as Taiwanese cities and mayors. In finding pairs of acronyms and full names of federal agencies, the precision at top 100 is nearly perfect (97%). The results for finding pairs of car makers and countries is good as well, with a high precision of 90%. For the last two datasets, we believe that MAP could be improved by increasing the number of bootstrapping iterations. At the end of this chapter, we include some example wrappers constructed and instances extracted using our proposed character-based relational wrappers.

7.4 Comparison to Prior Work

Extracting relations at character-level from semi-structured documents has been proposed [4, 21]. In particular, Brin's approach (DIPRE) is the most similar to Relational SEAL. He presented a technique which exploits the duality between sets of patterns and relations to grow the target relation starting from a small sample. He evaluated his technique on extracting the relation of author name and book title from the web. On this particular task, his system extracted 15,257 unique books with 19 of the 20 randomly-sampled books being correct (95% precision) by starting with five seed pairs. As far as we know, he only verified the correctness of the book titles but not the author names and relations. The five seed pairs that he used are shown below:

Author Name	/	**Book Title**
Isaac Asimov	/	The Robots of Dawn
David Brin	/	Startide Rising
James Gleick	/	Chaos: Making a New Science
Charles Dickens	/	Great Expectations
William Shakespeare	/	The Comedy of Errors

One difference between his and our approach is that his approach requires maximally-long contextual strings to bracket *all* seed occurrences. This technique has been experimentally illustrated to perform worse than SEAL's approach on unary relations as shown in Table 2.9. To compare against his approach, we input the first two seed pairs listed above (i.e., "*Isaac Asimov / The Robots of Dawn*" and "*David Brin / Startide Rising*") into Relational SEAL, performed ten bootstrapping iterations (took about 3 minutes), and obtained 26,000 author name and book title instance pairs, of which all extracted pairs in top 100 are correct (including book names, author names, and relations). We randomly sampled 100 extracted pairs from the top 15,257 pairs (i.e., the size of Brin's results) and found that they are all correct. In the next page, we show some top instance pairs extracted by Relational SEAL on the relation of author name and book title.

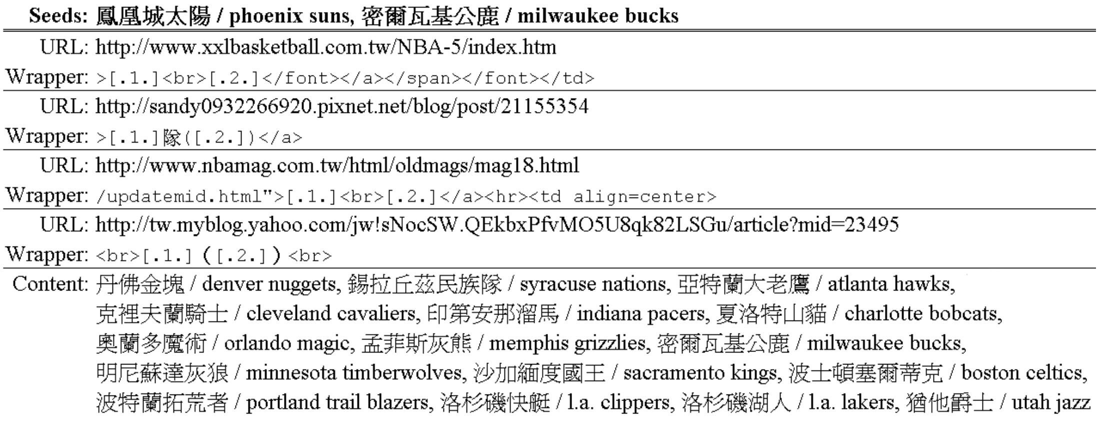
7.5 Summary

In this chapter, we introduce a system called Relational SEAL that utilizes an additional middle context for constructing relational wrappers. We showed that our relational set expansion approach is language-independent; it can be applied to non-English and even cross-lingual seeds and documents. In addition, we showed that our approach is effective on five relational datasets. Lastly, we illustrated that our bootstrapping approach described in Section 4.2.2.2 is also effective on expanding relational instances. At the end of this chapter, we show some real examples of relational wrappers constructed and candidate instances extracted.

#	Author Name / Book Title
1	David Brin / Startide Rising
2	David Brin / The Uplift War
3	Larry Niven / Ringworld
4	Dan Simmons / Hyperion
5	Orson Scott Card / Speaker for the Dead
6	Joe Haldeman / The Forever War
7	Isaac Asimov / Foundation's Edge
8	Frederik Pohl / Gateway
9	Isaac Asimov / The Gods Themselves
10	Arthur C. Clarke / Rendezvous with Rama
11	Dan Simmons / The Fall of Hyperion
12	Lois McMaster Bujold / Mirror Dance
13	William Gibson / Neuromancer
14	Orson Scott Card / Ender's Game
15	Neal Stephenson / The Diamond Age
16	Lois McMaster Bujold / Barrayar
17	Isaac Asimov / The Robots of Dawn
18	Kim Stanley Robinson / Green Mars
19	Kim Stanley Robinson / Blue Mars
20	Lois McMaster Bujold / The Vor Game
21	Connie Willis / Doomsday Book
22	Frank Herbert / Dune
23	Joe Haldeman / Forever Peace
24	David Brin / The Postman
25	Roger Zelazny / Lord of Light
26	Connie Willis / To Say Nothing of the Dog
27	Vernor Vinge / A Fire Upon the Deep
28	Kate Wilhelm / Where Late the Sweet Birds Sang
29	Philip K. Dick / The Man in the High Castle
30	Vernor Vinge / A Deepness in the Sky
31	James Blish / A Case of Conscience
32	Ursula K. Le Guin / The Dispossessed
33	Arthur C. Clarke / The Fountains of Paradise
34	Dan Simmons / The Rise of Endymion
35	Larry Niven / The Integral Trees
36	Ursula K. Le Guin / The Left Hand of Darkness
37	Joan D. Vinge / The Snow Queen
38	Robert A. Heinlein / Starship Troopers
39	Fritz Leiber / The Big Time
40	Gene Wolfe / The Claw of the Conciliator
41	John Brunner / Stand on Zanzibar
42	Greg Bear / Moving Mars
43	Frank Herbert / Children of Dune
44	William Gibson / Mona Lisa Overdrive
45	Fritz Leiber / The Wanderer
46	Robert A. Heinlein / Stranger in a Strange Land
47	Robert A. Heinlein / The Moon is a Harsh Mistress
48	Kim Stanley Robinson / Red Mars
49	John Varley / Titan
50	Robert A. Heinlein / Double Star
51	David Brin / Earth
52	Daniel Keyes / Flowers for Algernon
53	Neal Stephenson / Cryptonomicon
54	Isaac Asimov / The Caves of Steel
55	Stephen Baxter / The Time Ships
56	Alfred Bester / The Demolished Man
57	Lois McMaster Bujold / Memory
58	Lois McMaster Bujold / Falling Free
59	William Gibson / Count Zero
60	C. J. Cherryh / Cyteen
61	David Brin / Brightness Reef
62	Larry Niven / Protector
63	Vernor Vinge / Marooned in Realtime
64	Robert Silverberg / Lord Valentine's Castle
65	Frederik Pohl / Man Plus
66	Frederik Pohl / Beyond the Blue Event Horizon
67	Alexei Panshin / Rite of Passage
68	Ursula K. Le Guin / The Lathe of Heaven

Seeds:	**kentucky / steve beshear, north dakota / john hoeven**
URL:	http://wikifoia.pbworks.com/Alaska-Governor-Sarah-Palin
Wrapper:	Governor [.2.]">[.1.] Governor
URL:	http://blogs.suntimes.com/sweet/2008/02/sweet_state_dinner_for_governo.html
Wrapper:	<br /> <br /> The Honorable [.2.], Governor of [.1.] <br /> <br />
URL:	http://en.wikipedia.org/wiki/United_States_Senate_elections,_2010
Wrapper:	" title="Governor of [.1.]">Governor</a> <a href="/wiki/[.2.]" title="
URL:	http://ballotbox.governing.com/2008/07/index.html
Wrapper:	, [.1.]'s [.2.],
Content:	alabama / bob riley, alaska / sarah palin, arizona / janet napolitano, arkansas / mike huckabee, california / arnold schwarzenegger, colorado / bill ritter, connecticut / mary jodi rell, delaware / ruth ann minner, florida / charlie crist, georgia / sonny perdue, hawaii / linda lingle, idaho / butch otter...

Seeds:	**cia / central intelligence agency, usps / united states postal service**
URL:	http://www1.american.edu/dccampus/links/whitehouse.html
Wrapper:	<a href="http://www.[.1.].gov" class="Links2nd">[.2.]</a><span class="Links2nd">
URL:	http://www.usembassy.at/en/us/gov.htm
Wrapper:	/" target="_blank">[.2.] ([.1.])</a> -
URL:	http://www.nationmaster.com/encyclopedia/List-of-United-States-federal-agencies
Wrapper:	The [.2.] ([.1.]) is
URL:	http://www.nationmaster.com/encyclopedia/List-of-United-States-federal-agencies
Wrapper:	</li> <li>[.1.]- <a href="/encyclopedia/[.2.]" onmouseover="pv(event, 2
Content:	achp / advisory council on historic preservation, arc / appalachian regional commission, cftc / commodity futures trading commission, cia / central intelligence agency, cms / centers for medicare and medicaid services, exim bank / export import bank of the united states, ntrc / national transportation research center...



Seeds:	**鳳凰城太陽 / phoenix suns, 密爾瓦基公鹿 / milwaukee bucks**
URL:	http://www.xxlbasketball.com.tw/NBA-5/index.htm
Wrapper:	>[.1.] [.2.]</font></a></span></font></td>
URL:	http://sandy0932266920.pixnet.net/blog/post/21155354
Wrapper:	>[.1.]隊([.2.])</a>
URL:	http://www.nbamag.com.tw/html/oldmags/mag18.html
Wrapper:	/updatemid.html">[.1.] [.2.]</a><hr><td align=center>
URL:	http://tw.myblog.yahoo.com/jw!sNocSW.QEkbxPfvMO5U8qk82LSGu/article?mid=23495
Wrapper:	 [.1.]（[.2.]）
Content:	丹佛金塊 / denver nuggets, 錫拉丘茲民族隊 / syracuse nations, 亞特蘭大老鷹 / atlanta hawks, 克裡夫蘭騎士 / cleveland cavaliers, 印第安那溜馬 / indiana pacers, 夏洛特山貓 / charlotte bobcats, 奧蘭多魔術 / orlando magic, 孟菲斯灰熊 / memphis grizzlies, 密爾瓦基公鹿 / milwaukee bucks, 明尼蘇達灰狼 / minnesota timberwolves, 沙加緬度國王 / sacramento kings, 波士頓塞爾蒂克 / boston celtics, 波特蘭拓荒者 / portland trail blazers, 洛杉磯快艇 / l.a. clippers, 洛杉磯湖人 / l.a. lakers, 猶他爵士 / utah jazz

Chapter 8

Related Work

Comprehensive and accurate class-instance information has been proven useful in many applications, including relation learning from the web [5, 15, 31], feature generation for concept-learning [9] and co-reference resolution [23], dictionary construction for named entity recognition [2, 11, 25, 39, 41], query refinement in web search [26], enhancement of information retrieval [32] and list-type answers [50] for question answering, extension of WordNet [38, 48], "pseudo-users" construction for collaborative filtering [10], and similarity computation between attribute values in autonomous databases [52].

As mentioned in Chapter 1, most research on class instance acquisition has been conducted using evidence from either semi-structured documents, free text, or a combination of both. In this chapter, we examine some of the research conducted using each of these types of copora.

8.1 Semi-Structured Documents

Semi-structured documents are texts that do not conform with the formal structure of tables and data models associated with databases but contain tags or other markers to separate semantic elements and hierarchies of records and fields within the text, such as HTML, XML, tab-separated, and comma-separated text. In this thesis, we have developed a novel graph-based set expansion system that exploits semi-structured characteristics of web documents at character-level [46] and have shown that semi-structured documents provide more evidence and information than free text for discovering class instances. In this section, we present other research work that has also exploit semi-structured documents. Note that these techniques are more complex and less general than our SEAL approach (e.g., they parse HTML web pages, identify lists and tables, and filter out noisy tables).

Google Sets [44] is a well-known example of a web-based set expansion system; unfortunately, Google Sets is a proprietary method that may be changed at any time, so research results based on Google Sets cannot be reliably replicated. Google Sets contains a list identifier, a list classifier and a list processor. The list identifier identifies existing lists by a HTML tag (e.g., `<UL>`, `<OL>`, `<DL>`, `<H1>-<H6>` tags), by items placed in a table, items separated by commas or semicolons, or items separated by tabs. The list classifier generates an on-topic model and determine confidence scores that the existing lists were generated using the on-topic model. The list processor forms a list from the items in the existing lists and the determined confidence scores associated with the existing lists. Since Google Sets is publicly available, we used it as a baseline system for comparing against the basic SEAL in Section 2.5.6 and the Iterative SEAL in Section 4.4, and show that it performs worse than both versions of SEAL.

Etzioni *et al.* [15] present the KnowItAll system which contains a List Extractor (LE) component that is functionally similar to Google Sets and SEAL. The system uses an HTML parser for identifying sub-trees of a parsed web page. For each selected sub-tree, it finds one contextual pattern that maximally matches all of the seeds. However, SEAL does not require any parsing, and it finds all contextual patterns in the whole document that maximally match at least one instance of every seed. Etzioni *et al.* describe a number of possible variants of the LE component, but it is not clear which variant was used in their experiment. They manually evaluated their system and reported precisions and extraction sizes on their three sample problems, which are *cities* (52% of 151,016), *films* (72% of 78,859), and *scientists* (64% of 15,907).

Nadeau *et al.* [25] present a wrapper-based set-expansion system that identifies the location of specific class of instances within a HTML web page when given some seed instances. For example, a wrapper for identifying the location of car brands on a web page might contain the rule: "A brand is an HTML node of type `<a>`, with text length between 10 and 30 characters, in a table of depth 5 and with at least 3 other nodes in the page that satisfy the same rule." These constructed wrappers are then used as features for training a classifier to identify the locations of other positive instances. By utilizing these instances, they show that it is feasible to train an effective named-entity recognizer for extracting additional instances from free text.

Cafarella *et al.* [6] describe a system called WebTables that extracted 14.1 billion HTML tables (i.e., those embedded by the `<table>` tag) from Google's general-purpose web crawl, and used statistical classification techniques to find the estimated 154 million that contain high-quality relational data and class-instance information. They illustrate an approach to provide search-engine-style access to this huge volume of structured data and its applications – including schema auto-complete, attribute synonym finding, and join-graph discovery.

Suchanek *et al.* [40] present a system named YAGO, an ontological system that builds on entities and relations, including the hypernym hierarchy as well as non-taxonomic relations between entities (e.g., hasWonPrize). The facts have been automatically extracted from Wikipedia and unified with WordNet, using a carefully designed combination of rule-based and heuristic methods. They downloaded the English version of Wikipedia comprising of 1.6 million articles. The majority of Wikipedia pages have been automatically assigned to one or more classes. For example, the page about "*Albert Einstein*" is in the following classes: German Language Philosophers, Swiss Physicists, and 34 more. They reported a precision of about 95% on the correctness of their extracted facts, but they did not report the recall. Since they focus on only one web site on the web (i.e., Wikipedia), we believe that their recall would not be high.

8.2 Unstructured Documents

Unstructured documents (or free text) are text that have no meta-data and structure at all; such as newswire articles, conference publications, forum postings, web blogs, search engine query logs, and web pages stripped of HTML tags. Although we claim that our set expansion approach works on semi-structured document, we have often observed that the contexts in our constructed wrappers contain free text as well. Some examples of free-text wrappers are shown in Section 2.3.3. We estimated that these free-text wrappers consist of 10-20% of all wrappers constructed by our proposed approach. This observation shows that our approach is capable of constructing wrappers from unstructured documents as well. In this section, we are going to examine several research works utilizing free text to extract class instances. Note that many of these work utilize language-specific evidence and tools to pre-process their free-text corpora.

Talukdar *et al.* [41] present a context pattern induction method for named entity extraction. They extracted a fixed number N (context window size) of tokens immediately preceding and following the seed instances in their unlabeled data. From these extracted contexts, their system automatically selects *trigger words* to mark the beginning of a contextual pattern, which is then used for bootstrapping from free text. By using this method, they extended several classes of seed entity lists into larger lists. They evaluated their approach on a newswire corpus which contains 31 million documents and showed improvement in accuracy of a conditional random field-based named-entity tagger. As illustrated in Section 2.6.1, their system achieved a precision of 85.7% on the top 42 extracted instances on watch brand names using 17 seeds. SEAL achieved a precision of 100% at rank 42 by using only the first three of their 17 seeds, as presented in Table 2.11.

Kozareva *et al* [20] illustrated an approach that uses a single hyponym pattern (i.e., CLASS_NAME *such as* INSTANCE *and* $\star$) combined with graph structures to learn semantic class from search engine snippets. Their approach begins with just a class name and one seed instance and then automatically generate a ranked list of new instances. In their extractor, for proper instance names, they extract all capitalized words that immediately follow their learned patterns, but for common noun instances, they extract just one word if it is not capitalized. Section 5.5.1 shows that our ASIA approach outperforms theirs for all four datasets that they reported; however, their system requires more information, as it uses the name of the semantic class and a seed instance while ASIA requires only the class name. In terms of system runtime, for each semantic class, they reported that their extraction process usually finished overnight; however, ASIA usually finished within a minute by running on a single CPU machine.

Pasca [28] illustrated a set expansion approach that extracts instances from web search queries given a set of input seed instances, which is the same interface as SEAL. They show that search queries are highly valuable resource for web-based named entity discovery. Their extraction method consists of five stages: identification of query templates that match the seed instances; identification of candidate instances; internal representation of candidate instances and seed instances; and instance ranking.

The input to their experiments is a random sample of around 50 million unique, fully-anonymized queries in English submitted by users to the Google search engine in 2006. They evaluated their approach on 10 target classes, where each target class is specified through five seed instances. We evaluated our ASIA approach on those 10 classes, where each class is specified through only its class name. Section 5.5.2 shows that ASIA outperforms their system, with an average precision of 100% at rank 25 (vs. their 98%) and 93% at rank 250 (vs. their 80%).

Van Durme and Paşca [14] present a method for extracting large numbers of semantic classes along with their corresponding instances, based on the recombination of elements clustered through distributional similarity. The input to their algorithm is a large collection of pairs of class names and instances. They claim that this collection can be extracted using pattern-based methods such as those presented by Hearst [17]. Their experiments relied on the unstructured text available within a collection of approximately 100 million web documents in English, as available in a web repository snapshot from 2006 maintained by Google. The documents were cleaned of HTML, tokenized, split into sentences, and part-of-speech tagged using the TnT tagger [3]. In Section 5.5.3, we compare ASIA against Van Durme's system and show that ASIA outperforms their system on five semantic classes randomly sampled by Talukdar *et al.* [42]. ASIA achieved an average precision of 99% while Van Durme's system achieved 87%.

Snow [38] propose an approach that incorporates evidence from multiple classifiers over heterogenous relationships to optimize the entire structure of semantic taxonomies, using knowledge of a word's coordinate terms to help in determining its hypernyms. They apply the algorithm on the problem of sense-disambiguated noun hyponym acquisition, where they combine the predictions of hypernym and coordinate term classifiers with the knowledge in a pre-existing semantic taxonomy – WordNet 2.1. They extended the WordNet by adding 10,000 entries (synsets) at a relatively high precision of 84%. They have made several versions of the extended WordNet available[1]. For comparison purposes, we selected the version (+30K) that achieved the best F-score (30.9%) in their experiments. The results presented in Section 5.5.4 show that instances extracted by ASIA has much higher precision (98% vs. 70%) and much higher relative recall (92% vs. 47%) than those in their extended WordNet.

[1] `http://ai.stanford.edu/~rion/swn/`

8.3 Combination of Both

In this section, we examine approaches that utilize evidence from both semi-structured and unstructured documents. Note that they all focus on documents written in the English language only.

Talukdar *et al.* [42] present a graph-based semi-supervised label propagation algorithm called Adsorption for acquiring open-domain labeled classes and their instances from a combination of unstructured and structured text sources. They construct a graph where each node represent either an instance or a class, and an edge exists between an instance node and a class node if the instance belongs to that class. The Adsorption label propagation algorithm is then applied to that graph to label all nodes based on the graph structure, ultimately producing a probability distribution over classes for each instance node. The unstructured text that they utilize was the web documents prepared and used by Van Durme & Paşca as described above, and the structured text was the WebTables [6] as also described above. Note that Adsorption requires inputs of a class name and several seed instances (five seeds per class were used in their experiments). In Section 5.5.3, we compare ASIA against their system on five sample classes used in their paper, and we show that ASIA outperforms their system, achieving an average precision of 99% while their system achieved 79%.

Carlson *et al.* [8] present an approach of using semi-supervised learning to extract instances of various classes (e.g., academic fields, athletes) and relations (e.g., "`<athlete>` plays `<sport>`") from semi-structured and unstructured text. This paper shows that a much higher accuracy in semi-supervised learning can be achieved by a learner they call Meta-Bootstrap Learner (MBL), which couples the simultaneous training of two extractors – a free-text pattern learner and our set-expansion system SEAL. MBL couples the training of multiple extractors using a multi-view constraint that requires them to agree. The input to MBL is a handful of labeled training examples of each category or relation, and the input to their pattern learner is 200 million unlabeled web pages. They pre-process the web pages by first parsing the HTML, and then filtering out non-English pages and pages containing stop-words, web spam, and adult content. The pages were then split into sentences, tokenized, and part-of-speech tagged.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

In this thesis, we have proposed a novel graph-based approach to set expansion using semi-structured documents at the character-level. Based on our proposed approach, we developed an open-source set-expansion system called Set Expander for Any Language (SEAL) that is independent of both human and mark-up language. Throughout this thesis, we have developed multiple variants of SEAL for various purposes. In this section, we will review each variant developed and its contributions.

The basic version of SEAL is composed of a fetcher, an extractor, and a ranker. We have shown that it is capable of handling various languages such as English, Chinese, and Japanese. By conducting ablation studies using SEAL, we have also shown that our random walk approach outperforms other state-of-the-art rankers for the problem of set expansion. The studies also show that our novel character-based wrapper induction technique is more effective than a simpler and more common technique. The experimental results illustrate that character-based wrappers are better suited than HTML-based wrappers for the task of set expansion.

The noise-resistant version of SEAL is capable of improving the quality of noisy answers generated by QA systems on list questions. It is developed through the following three changes to SEAL: 1) alter the fetcher so that it retrieves more documents, 2) modify the extractor so that it is more lenient on inducing wrappers, and 3) append hint words to search queries sent by the fetcher to retrieve more on-topic documents. In our experimental results, we have also shown that the Noise-Resistant SEAL outperforms Google Sets when given noisy seed instances (i.e., the top answers from QA systems).

The iterative version of SEAL (iSEAL) executes SEAL in multiple iterations where statistics are accumulated from iteration to iteration. By using iSEAL, we have shown that set expansion performance can be improved monotonically if we bootstrap the results using ISS (increasing seed size) and rank the results using random walk. By conducting ablation studies using iSEAL, we have also shown that in supervised mode, random walk is comparable to the state-of-the-art ranker, Bayesian Sets, but in bootstrap mode, random walk is the best due to its robustness to noisy seeds.

The Automatic Set Instance Acquirer (ASIA) is capable of extracting class instances in several languages given only the semantic class name. It utilizes three novel components: 1) the Noisy Instance Provider, which extracts noisy instances using Hearst patterns from search result snippets, 2) the Noisy Instance Expander, which expands the noisy instances using a better variant of the Lenient Extractor in the Noise-Resistant SEAL, and 3) the Bootstrapper, which bootstraps expanded instances using iSEAL. In the experimental results, we have shown that ASIA achieves higher precision and recall than many free-text systems (even though many of them require more inputs than a class name). These results provide evidence towards our conjecture that semi-structured documents provide more information than free text for discovering class instances.

The bilingual version of SEAL calls two iSEALs alternately to bootstrap instances of the same semantic class but in a different language. The instances extracted in different languages are bridged together by ANET – a simple named-entity translator. By using Bilingual SEAL, we have shown that the performance of set expansion can be improved by exploiting redundant information of classes in different languages. In addition, the precision of translation pairs generated as by-products from the bilingual approach has also been improved dramatically.

The relational version of SEAL is developed to expand binary relations by adding a middle context to the wrappers induced by the Extractor of SEAL. Similar to SEAL, Relational SEAL is independent of both human and mark-up language. The experimental results indicate that its character-based approach is effective and also performs better than HTML-based methods. The results also illustrate that Relational SEAL can be bootstrapped to improve its expansion results, using the same bootstrapping technique implemented in iSEAL.

9.2 Future Work

In this thesis, the instances extracted by SEAL were never automatically validated for their correctness. In the future work, we propose to examine these instances based on their distributional similarity [22] in free text. For example, the sentences "*Clinton* vetoed the bill" and "*Bush* vetoed the bill" suggest that *Clinton* and *Bush* may be semantically related. We will use this measure to determine if all instances are coherent to the expanded set, and if not, we will filter out those that are least coherent.

The above described approach may also be used to partition the expanded instances into subclasses, or clusters. For example, having an expanded set of car brands, we can automatically partition it into clusters such as Japanese cars, luxurious cars, and affordable cars. This can be achieved by using the contexts of the instances as features and grouping together the instances that tend to appear in similar contexts.

Automatic identification of concept names given some example instances (the opposite task of that solved by ASIA) may be useful in some applications. For example, given "*Honda*", "*Nissan*", and "*Toyota*", the probable semantic class names of these instances include "*Car Makers*", "*Car Manufacturers*", "*Car Brands*", "*Affordable Cars*", and "*Japanese Cars*". These class names can be either discovered through meta-data in web pages (e.g., list and table headers) or lexical evidence in free text (e.g., "`CLASS_NAME` such as *Honda*" suggest that `CLASS_NAME` is a probable hypernym of *Honda*).

As mentioned in Section 5.5.2, ASIA sometimes output subclass names rather than instance names (e.g., given "people", it outputs "writers", "actors", and "athletes"). One future work would be to identify subclass names from instance names. A simple approach would be to identify subclasses based on capitalization, since instances names are usually (capitalized) named entities while class names are not. A more complex approach would be based on the occurrence frequency of some lexical patterns (e.g., "*actors* such as" occurs frequently while "*Tom Cruise* such as" does not).

If we combine all the abovementioned work with the work of this thesis, we can have a system that automatically constructs a hypernym hierarchy. The procedure goes as follows: Given a semantic class, we let ASIA extract all possible instances first. We then partition these instances into clusters and assign each cluster its most probable class name. Lastly, we repeat these procedures for every such class name until no more clusters could be generated.

References

[1] Eugene Agichtein and Luis Gravano. **Snowball: Extracting Relations from Large Plain-Text Collections**. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, pages 85–94, 2000. 96

[2] Joohui An, Seungwoo Lee, and Gary Geunbae Lee. **Automatic acquisition of named entity tagged corpus from world wide web**. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 165–168, Morristown, NJ, USA, 2003. Association for Computational Linguistics. 1, 105

[3] T. Brants. **TnT – a statistical part-of-speech tagger**, 2000. 78, 109

[4] Sergey Brin. **Extracting Patterns and Relations from the World Wide Web**. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT98*, pages 172–183, 1998. 95, 101

[5] Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. **KnowItNow: fast, scalable information extraction from the web**. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 563–570, Morristown, NJ, USA, 2005. Association for Computational Linguistics. 1, 105

[6] Michael J. Cafarella, Alon Halevy, Daisy Z. Wang, Eugene Wu, and Yang Zhang. **WebTables: exploring the power of tables on the web**. *Proceedings of the VLDB Endowment*, **1**(1):538–549, 2008. 2, 18, 78, 96, 107, 110

[7] A. Carlson, J. Betteridge, E.R. Hruschka Junior, and T.M. Mitchell. **Coupling Semi-Supervised Learning of Categories and Relations**. In *NAACL HLT Workshop on Semi-supervised Learning for Natural Language Processing*, pages 1–9. Association for Computational Linguistics, 2009. 96

[8] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr., and T. M. Mitchell. **Coupled Semi-Supervised Learning for Information Extraction**. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM)*, 2010. 2, 110

[9] William W. Cohen. **Automatically Extracting Features for Concept Learning from the Web**. In Pat Langley, editor, *ICML*, pages 159–166. Morgan Kaufmann, 2000. 1, 105

[10] William W. Cohen and Wei Fan. **Learning Page-Independent Heuristics for Extracting Data from Web Pages**. *Computer Networks*, **31**(11-16):1641–1652, 1999. 1, 105

[11] William W. Cohen and Sunita Sarawagi. **Exploiting dictionaries in named entity extraction: combining semi-Markov extraction processes and data integration methods**. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *KDD*, pages 89–98. ACM, 2004. 1, 105

[12] H.T. Dang, D. Kelly, and J. Lin. **Overview of the TREC 2007 Question Answering Track**. *Proceedings of the Sixteenth Text REtrieval Conference*, 2007. 35, 36

[13] H.T. Dang, J. Lin, and D. Kelly. **Overview of the TREC 2006 Question Answering Track**. *Proceedings of the Fifteenth Text REtrieval Conference*, 2006. 35, 36

[14] Benjamin Van Durme and Marius Pasca. **Finding Cars, Goddesses and Enzymes: Parametrizable Acquisition of Labeled Instances for Open-Domain Information Extraction**. In Dieter Fox and Carla P. Gomes, editors, *AAAI*, pages 1243–1248. AAAI Press, 2008. 2, 3, 78, 81, 109

[15] Oren Etzioni, Michael J. 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**. *Artificial Intelligence*, **165**(1):91–134, 2005. 1, 2, 7, 18, 25, 47, 50, 63, 66, 67, 72, 96, 105, 106

[16] Zoubin Ghahramani and Katherine A. Heller. **Bayesian Sets**. In *NIPS*, 2005. 2, 22, 23, 31, 32, 48

[17] Marti A. Hearst. **Automatic acquisition of hyponyms from large text corpora**. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992. 2, 63, 66, 67, 78, 109

[18] J. Ko, L. Si, and E. Nyberg. **A Probabilistic Framework for Answer Selection in Question Answering**. *Proceedings of NAACL-HLT*, 2007. 36

[19] Zornitsa Kozareva. **Bootstrapping Named Entity Recognition with Automatically Generated Gazetteer Lists**. In *EACL*. The Association for Computer Linguistics, 2006. 47, 72

[20] Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. **Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs**. In *Proceedings of ACL-08: HLT*, pages 1048–1056, Columbus, Ohio, June 2008. Association for Computational Linguistics. 2, 3, 63, 64, 66, 67, 75, 81, 108

[21] N. Kushmerick, D. Weld, and B. Doorenbos. **Wrapper induction for information extraction**. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1997. 95, 101

[22] Dekang Lin and Patrick Pantel. **Concept Discovery from Text**. In *In Proceedings of Conference on Computational Linguistics*, pages 577–583, 2002. 113

[23] J Mccarthy and Wendy G. Lehnert. **Using Decision Trees for Coreference Resolution**. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1050–1055, 1995. 1, 105

[24] Donald R. Morrison. **PATRICIA–Practical Algorithm To Retrieve Information Coded in Alphanumeric**. *J. ACM*, **15**(4):514–534, October 1968. 13

[25] David Nadeau, Peter D. Turney, and Stan Matwin. **Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity**. In Luc Lamontagne and Mario Marchand, editors, *Canadian Conference on AI*, **4013** of *Lecture Notes in Computer Science*, pages 266–277. Springer, 2006. 1, 2, 18, 25, 47, 50, 72, 96, 105, 106

[26] Marius Paşca. **Acquisition of categorized named entities for web search**. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 137–145, New York, NY, USA, 2004. ACM. 1, 2, 63, 66, 67, 105

[27] Marius Paşca. **Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds**. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 101–110, New York, NY, USA, 2007. ACM. 2, 66

[28] Marius Paşca. **Weakly-supervised discovery of named entities using web search queries**. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 683–690, New York, NY, USA, 2007. ACM. 2, 3, 66, 76, 81, 108

[29] Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. **Organizing and searching the world wide web of facts – step one: the one-million fact extraction challenge**. In *AAAI'06: proceedings of the 21st national conference on Artificial intelligence*, pages 1400–1405. AAAI Press, 2006. 2, 96

[30] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. **The PageRank Citation Ranking: Bringing Order to the Web**. Technical report, Stanford Digital Library Tech. Project, 1998. 21, 22, 48

[31] Patrick Pantel and Marco Pennacchiotti. **Espresso: leveraging generic patterns for automatically harvesting semantic relations**. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 113–120, Morristown, NJ, USA, 2006. Association for Computational Linguistics. 1, 96, 105

[32] Patrick Pantel and Deepak Ravichandran. **Automatically Labeling Semantic Classes**. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 321–328, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics. 1, 2, 63, 66, 67, 105

[33] John M. Prager, Jennifer Chu-Carroll, and Krzysztof Czuba. **Question Answering Using Constraint Satisfaction: QA-By-Dossier-With-Contraints**. In *ACL*, pages 574–581, 2004. 3, 21

[34] Ellen Riloff and Rosie Jones. **Learning dictionaries for information extraction by multi-level bootstrapping**. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 474–479, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence. 2

[35] N. Schlaefer, P. Gieselmann, and G. Sautter. **The Ephyra QA System at TREC 2006**. *Proceedings of the Fifteenth Text REtrieval Conference*, 2006. 36

[36] N. Schlaefer, G. Sautter, J. Ko, J. Betteridge, M. Pathak, and E. Nyberg. **Semantic Extensions of the Ephyra QA System in TREC 2007**. *Proceedings of the Sixteenth Text REtrieval Conference*, 2007. 35, 36

[37] Burr Settles. **Biomedical Named Entity Recognition using Conditional Random Fields and Rich Feature Sets**. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko, editors, *COLING 2004 International Joint workshop on NLPBA/BioNLP*, pages 107–110, Geneva, Switzerland, August 28th and 29th 2004. COLING. 3, 21

[38] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. **Semantic taxonomy induction from heterogenous evidence**. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 801–808, Morristown, NJ, USA, 2006. Association for Computational Linguistics. 1, 2, 3, 64, 66, 79, 81, 96, 105, 109

[39] Mark Stevenson and Robert Gaizauskas. **Using corpus-derived name lists for named entity recognition**. In *Proceedings of the sixth conference on Applied natural language processing*, pages 290–295, Morristown, NJ, USA, 2000. Association for Computational Linguistics. 1, 105

[40] Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. **YAGO: A Core of Semantic Knowledge - Unifying WordNet and Wikipedia**. In Carey L. Williamson, Mary Ellen Zurko, and Prashant J. Patel-Schneider, Peter F. Shenoy, editors, *16th International World Wide Web Conference (WWW 2007)*, pages 697–706, Banff, Canada, 2007. ACM. 2, 107

[41] Partha P. Talukdar, Thorsten Brants, Mark Liberman, and Fernando Pereira. **A Context Pattern Induction Method for Named Entity Extraction**. In *Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, 2006. 1, 3, 31, 96, 105, 108

[42] Partha Pratim Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. **Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks**. In *EMNLP*, pages 582–590. ACL, 2008. 2, 3, 18, 78, 81, 109, 110

[43] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. **Fast Random Walk with Restart and Its Applications**. In *ICDM*, pages 613–622. IEEE Computer Society, 2006. 10, 19, 48, 71, 85

[44] Simon Tong and Jeff Dean. **System and methods for automatically creating lists**, 03 2008. 3, 21, 96, 106

[45] Jinghua Wang, Jianyi Liu, and Cong Wang. **Keyword Extraction Based on PageRank**. In Zhi-Hua Zhou, Hang Li, and Qiang Yang, editors, *PAKDD*, **4426** of *Lecture Notes in Computer Science*, pages 857–864. Springer, 2007. 21

[46] Richard C. Wang and William W. Cohen. **Language-Independent Set Expansion of Named Entities Using the Web**. In *ICDM*, pages 342–350. IEEE Computer Society, 2007. 3, 105

[47] Richard C. Wang and William W. Cohen. **Iterative Set Expansion of Named Entities Using the Web**. In *ICDM*, pages 1091–1096. IEEE Computer Society, 2008. 4

[48] Richard C. Wang and William W. Cohen. **Automatic Set Instance Extraction using the Web**. In *ACL-IJCNLP*, Suntec City, Singapore, August 2009. 1, 4, 105

[49] Richard C. Wang and William W. Cohen. **Character-level Analysis of Semi-Structured Documents for Set Expansion**. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1503–1512, Singapore, August 2009. Association for Computational Linguistics. 4

[50] Richard C. Wang, Nico Schlaefer, William W. Cohen, and Eric Nyberg. **Automatic Set Expansion for List Question Answering**. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 947–954, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. 1, 4, 35, 105

[51] 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, Morristown, NJ, USA, 2002. Association for Computational Linguistics. 2

[52] Garrett Wolf, Hemal Khatri, Yi Chen, and Subbarao Kambhampati. **QUIC: A System for Handling Imprecision & Incompleteness in Autonomous Databases (Demo)**. In *CIDR*, pages 263–268. www.crdrdb.org, 2007. 1, 105

[53] Christopher C. Yang and K. Y. Chan. **Retrieving multimedia web objects based on PageRank algorithm**. In Allan Ellis and Tatsuya Hagino, editors, *WWW*, pages 906–907. ACM, 2005. 21

[54] Hugo Zaragoza, Henning Rode, Peter Mika, Jordi Atserias, Massimiliano Ciaramita, and Giuseppe Attardi. **Ranking very many typed entities on wikipedia**. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 1015–1018, New York, NY, USA, 2007. ACM. 2

Appendix A

The 36 Unary Datasets

English Dataset: disney-movies

(1) Snow White And The Seven Dwarfs (2) Pinocchio (3) Fantasia (4) Dumbo (5) Bambi (6) Saludos Amigos (7) The Three Caballeros (8) Make Mine Music (9) Fun And Fancy Free (10) Melody Time (11) The Adventures Of Ichabod And Mr. Toad, Adventures Of Ichabod And Mr. Toad (12) Cinderella (13) Alice In Wonderland (14) Peter Pan (15) Lady And The Tramp (16) Sleeping Beauty (17) 101 Dalmatians, One Hundred and One Dalmatians (18) The Sword In The Stone, Sword In The Stone (19) The Jungle Book, Jungle Book (20) The Aristocats, Aristocats (21) Robin Hood (22) The Many Adventures Of Winnie The Pooh, Winnie the Pooh (23) The Rescuers, Rescuers (24) The Fox And The Hound, Fox And The Hound (25) The Black Cauldron, Black Cauldron (26) The Great Mouse Detective (27) Oliver & Company, Oliver And Company (28) The Little Mermaid, Little Mermaid (29) The Rescuers Down Under, Rescuers Down Under (30) Beauty And The Beast (31) Aladdin (32) The Lion King, Lion King (33) Pocahontas (34) The Hunchback Of Notre Dame, Hunchback Of Notre Dame (35) Hercules (36) Mulan (37) Tarzan (38) Fantasia 2000, Fantasia/2000 (39) The Emperor's New Groove, Emperor's New Groove (40) Atlantis: The Lost Empire (41) Lilo & Stitch, Lilo And Stitch (42) Treasure Planet (43) Brother Bear (44) Home On The Range (45) Chicken Little (46) Meet the Robinsons (47) American Dog (48) Rapunzel Unbraided (49) The Frog Princess

English Dataset: constellations

(1) Andromeda (2) Antlia (3) Apus (4) Aquarius (5) Aquila (6) Ara (7) Aries (8) Auriga (9) Bootes, Bootës (10) Caelum (11) Camelopardalis (12) Cancer (13) Canes Venatici (14) Canis Major (15) Canis Minor (16) Capricornus, Capricorn (17) Carina (18) Cassiopeia (19) Centaurus (20) Cepheus (21) Cetus (22) Chamaeleon (23) Circinus (24) Columba (25) Coma Berenices (26) Corona Australis (27) Corona Borealis (28) Corvus (29) Crater (30) Crux (31) Cygnus (32) Delphinus (33) Dorado (34) Draco (35) Equuleus (36) Eridanus (37) Fornax (38) Gemini (39) Grus (40) Hercules (41) Horologium (42) Hydra (43) Hydrus (44) Indus (45) Lacerta (46) Leo (47) Leo Minor (48) Lepus (49) Libra (50) Lupus (51) Lynx (52) Lyra (53) Mensa (54) Microscopium (55) Monoceros (56) Musca (57) Norma (58) Octans (59) Ophiuchus (60) Orion (61) Pavo (62) Pegasus (63) Perseus (64) Phoenix (65) Pictor (66) Pisces (67) Piscis Austrinus (68) Puppis (69) Pyxis (70) Reticulum (71) Sagitta (72) Sagittarius (73) Scorpius, Scorpio (74) Sculptor (75) Scutum (76) Serpens (77) Sextans (78) Taurus (79) Telescopium (80) Triangulum (81) Triangulum Australe (82) Tucana (83) Ursa Major (84) Ursa Minor (85) Vela (86) Virgo (87) Volans (88) Vulpecula

English Dataset: countries

(1) Afghanistan (2) Albania, Republic of Albania (3) Algeria, People's Democratic Republic of Algeria (4) Andorra, Principality of Andorra (5) Angola, Republic of Angola (6) Antigua and Barbuda, Antigua & Barbuda (7) Argentina, Argentine Republic (8) Armenia, Republic of Armenia (9) Australia, Commonwealth of Australia (10) Austria, Republic of Austria (11) Azerbaijan, Republic of Azerbaijan (12) Bahamas, The Bahamas, Commonwealth of The Bahamas (13) Bahrain, Kingdom of Bahrain (14) Bangladesh, People's Republic of Bangladesh (15) Barbados (16) Belarus, Republic of Belarus (17) Belgium, Kingdom of Belgium (18) Belize (19) Benin, Republic of Benin (20) Bhutan, Kingdom of Bhutan (21) Bolivia, Republic of Bolivia (22) Bosnia and Herzegovina, Bosnia & Herzegovina (23) Botswana, Republic of Botswana (24) Brazil, Federative Republic of Brazil, Brasil (25) Brunei, Negara Brunei Darussalam, Brunei Darussalam (26) Bulgaria, Republic of Bulgaria (27) Burkina Faso (28) Burundi, Republic of Burundi (29) Cambodia, Kingdom of Cambodia (30) Cameroon, Republic of Cameroon (31) Canada (32) Cape Verde, Republic of Cape Verde (33) Central African Republic (34) Chad, Republic of Chad (35) Chile, Republic of Chile (36) China, People's Republic of China, PRC (37) Colombia, Republic of Colombia (38) Comoros, Union of the Comoros (39) Congo, Democratic Republic of the Congo, Zaire (40) Congo, Republic of the Congo (41) Costa Rica, Republic of Costa Rica (42) Cote d'Ivoire, Côte d'Ivoire, Republic of Cote d'Ivoire, Republic of Côte d'Ivoire (43) Croatia, Republic of Croatia (44) Cuba, Republic of Cuba (45) Cyprus, Republic of Cyprus (46) Czech Republic (47) Denmark, Kingdom of Denmark (48) Djibouti, Republic of Djibouti (49) Dominica, Commonwealth of Dominica (50) Dominican Republic (51) Ecuador, Republic of Ecuador (52) East Timor, Timor-Leste, Democratic Republic of Timor-Leste (53) England (54) Egypt, Arab Republic of Egypt (55) El Salvador, Republic of El Salvador (56) Equatorial Guinea, Republic of Equatorial Guinea (57) Eritrea, State of Eritrea (58) Estonia, Republic of Estonia (59) Ethiopia, Federal Democratic Republic of Ethiopia (60) Fiji, Republic of the Fiji Islands (61)

Finland, Republic of Finland (62) France, French Republic (63) Gabon, Gabonese Republic (64) Gambia, The Gambia, Republic of The Gambia (65) Georgia (66) Germany, Federal Republic of Germany (67) Ghana, Republic of Ghana (68) Greece, Hellenic Republic (69) Grenada (70) Guatemala, Republic of Guatemala (71) Guinea, Republic of Guinea (72) Guinea-Bissau, Republic of Guinea-Bissau, Guinea Bissau (73) Guyana, Co-operative Republic of Guyana (74) Haiti, Republic of Haiti (75) Honduras, Republic of Honduras (76) Hungary, Republic of Hungary (77) Iceland, Republic of Iceland (78) India, Republic of India (79) Indonesia, Republic of Indonesia (80) Iran, Islamic Republic of Iran (81) Iraq, Republic of Iraq (82) Ireland (83) Israel, State of Israel (84) Italy, Italian Republic (85) Jamaica (86) Japan (87) Jordan, Hashemite Kingdom of Jordan (88) Kazakhstan, Republic of Kazakhstan (89) Kenya, Republic of Kenya (90) Kiribati, Republic of Kiribati (91) Korea, North Korea, Democratic People's Republic of Korea (92) Korea, South Korea, Republic of Korea (93) Kuwait, State of Kuwait (94) Kyrgyzstan, Kyrgyz Republic (95) Laos, Lao People's Democratic Republic (96) Latvia, Republic of Latvia (97) Lebanon, Republic of Lebanon (98) Lesotho, Kingdom of Lesotho (99) Liberia, Republic of Liberia (100) Libya, Great Socialist People's Libyan Arab Jamahiriya (101) Liechtenstein, Principality of Liechtenstein (102) Lithuania, Republic of Lithuania (103) Luxembourg, Grand Duchy of Luxembourg (104) Macedonia, Republic of Macedonia (105) Madagascar, Republic of Madagascar (106) Malawi, Republic of Malawi (107) Malaysia (108) Maldives, Republic of Maldives (109) Mali, Republic of Mali (110) Malta, Republic of Malta (111) Marshall Islands, Republic of the Marshall Islands (112) Mauritania, Islamic Republic of Mauritania (113) Mauritius, Republic of Mauritius (114) Mexico, United Mexican States (115) Micronesia, Federated States of Micronesia (116) Moldova, Republic of Moldova (117) Monaco, Principality of Monaco (118) Mongolia (119) Montenegro, Republic of Montenegro (120) Morocco, Kingdom of Morocco (121) Mozambique, Republic of Mozambique (122) Myanmar, Union of Myanmar, Burma (123) Namibia, Republic of Namibia (124) Nauru, Republic of Nauru (125) Nepal, State of Nepal (126) Netherlands, The Netherlands, Kingdom of the Netherlands (127) New Zealand (128) Nicaragua, Republic of Nicaragua (129) Niger, Republic of Niger (130) Nigeria, Federal Republic of Nigeria (131) Northern Ireland (132) Norway, Kingdom of Norway (133) Oman, Sultanate of Oman (134) Pakistan, Islamic Republic of Pakistan (135) Palau, Republic of Palau (136) Palestine, Palestinian State, State of Palestine (137) Panama, Republic of Panama (138) Papua New Guinea, Independent State of Papua New Guinea (139) Paraguay, Republic of Paraguay (140) Peru, Republic of Peru (141) Philippines, The Philippines, Republic of the Philippines (142) Poland, Republic of Poland (143) Portugal, Portuguese Republic (144) Qatar, State of Qatar (145) Romania (146) Russia, Russian Federation (147) Rwanda, Republic of Rwanda (148) Saint Kitts and Nevis, Saint Kitts & Nevis, St. Kitts & Nevis, St. Kitts and Nevis, Federation of Saint Christopher and Nevis (149) Saint Lucia, St. Lucia (150) Saint Vincent and the Grenadines, Saint Vincent & the Grenadines, St. Vincent & The Grenadines, St. Vincent and The Grenadines (151) Samoa, Independent State of Samoa (152) San Marino, Most Serene Republic of San Marino (153) Sao Tome and Principe, São Tomé and Príncipe, Democratic Republic of Sao Tome and Principe, Democratic Republic of São Tomé and Príncipe (154) Saudi Arabia, Kingdom of Saudi Arabia (155) Scotland (156) Senegal, Republic of Senegal (157) Serbia, Republic of Serbia (158) Seychelles, Republic of Seychelles (159) Sierra Leone, Republic of Sierra Leone (160) Singapore, Republic of Singapore (161) Slovakia, Slovak Republic (162) Slovenia, Republic of Slovenia (163) Solomon Islands (164) Somalia (165) South Africa, Republic of South Africa (166) Spain, Kingdom of Spain (167) Sri Lanka, Democratic Socialist Republic of Sri Lanka (168) Sudan, Republic of the Sudan (169) Suriname, Republic of Suriname (170) Swaziland, Kingdom of Swaziland (171) Sweden, Kingdom of Sweden (172) Switzerland, Swiss Confederation (173) Syria, Syrian Arab Republic (174) Taiwan, Republic of China, ROC (175) Tajikistan, Republic of Tajikistan (176) Tanzania, United Republic of Tanzania (177) Thailand, Kingdom of Thailand (178) Togo, Togolese Republic (179) Tonga, Kingdom of Tonga (180) Trinidad and Tobago, Trinidad & Tobago, Republic of Trinidad and Tobago (181) Tunisia, Tunisian Republic (182) Turkey, Republic of Turkey (183) Turkmenistan (184) Tuvalu (185) Uganda, Republic of Uganda (186) Ukraine (187) United Arab Emirates, Emirates, UAE (188) United Kingdom, United Kingdom of Great Britain and Northern Ireland, Britain, UK (189) United States, United States of America, America, USA, U.S. (190) Uruguay, Eastern Republic of Uruguay (191) Uzbekistan, Republic of Uzbekistan (192) Vanuatu, Republic of Vanuatu (193) Vatican City, State of the Vatican City, Holy See (194) Venezuela, Bolivarian Republic of Venezuela (195) Vietnam, Socialist Republic of Vietnam, Viet Nam (196) Wales (197) Western Sahara, Sahrawi Arab Democratic Republic (198) Yemen, Republic of Yemen (199) Zambia, Republic of Zambia (200) Zimbabwe, Republic of Zimbabwe

English Dataset: mlb-teams

(1) Atlanta Braves, braves (2) Florida Marlins, marlins (3) New York Mets, mets (4) Philadelphia Phillies, phillies (5) Washington Nationals, nationals (6) Chicago Cubs, cubs (7) Cincinnati Reds, reds (8) Houston Astros, astros (9) Milwaukee Brewers, brewers (10) Pittsburgh Pirates, pirates (11) St. Louis Cardinals, cardinals (12) Arizona Diamondbacks, diamondbacks (13) Colorado Rockies, rockies (14) Los Angeles Dodgers, dodgers (15) San Diego Padres, padres (16) San Francisco Giants, giants (17) Baltimore Orioles, orioles (18) Boston Red Sox, red sox (19) New York Yankees, yankees (20) Tampa Bay Devil Rays, Tampa Bay Rays, devil rays, rays (21) Toronto Blue Jays, blue jays (22) Chicago White Sox, white sox (23) Cleveland Indians, indians (24) Detroit Tigers, tigers (25) Kansas City Royals, royals (26) Minnesota Twins, twins (27) Los Angeles Angels of Anaheim, Los Angeles Angels, angels (28) Oakland Athletics, athletics (29) Seattle Mariners, mariners (30) Texas Rangers, rangers

English Dataset: nba-teams

(1) Boston Celtics (2) New Jersey Nets (3) New York Knicks (4) Philadelphia 76ers (5) Toronto Raptors (6) Chicago Bulls (7) Cleveland Cavaliers (8) Detroit Pistons (9) Indiana Pacers (10) Milwaukee Bucks (11) Atlanta Hawks (12) Charlotte Bobcats (13) Miami Heat (14) Orlando Magic (15) Washington Wizards (16) Dallas Mavericks (17) Houston Rockets (18) Memphis Grizzlies (19) New Orleans Hornets, New Orleans/Oklahoma City Hornets, Charlotte Hornets (20) San Antonio Spurs (21) Denver Nuggets (22) Minnesota Timberwolves (23) Portland Trail Blazers (24) Oklahoma City Thunder, Seattle Supersonics (25) Utah Jazz (26) Golden State Warriors (27) Los Angeles Clippers (28) Los Angeles Lakers (29) Phoenix Suns (30) Sacramento Kings

English Dataset: nfl-teams

(1) Buffalo Bills, bills (2) Miami Dolphins, dolphins (3) New England Patriots, patriots (4) New York Jets, jets (5) Baltimore Ravens, ravens (6) Cincinnati Bengals, bengals (7) Cleveland Browns, browns (8) Pittsburgh Steelers, steelers (9) Houston Texans, texans (10) Indianapolis Colts, colts (11) Jacksonville Jaguars, jaguars (12) Tennessee Titans, titans (13) Denver Broncos,

broncos (14) Kansas City Chiefs, chiefs (15) Oakland Raiders, raiders (16) San Diego Chargers, chargers (17) Dallas Cowboys, cowboys (18) New York Giants, giants (19) Philadelphia Eagles, eagles (20) Washington Redskins, redskins (21) Chicago Bears, bears (22) Detroit Lions, lions (23) Green Bay Packers, packers (24) Minnesota Vikings, vikings (25) Atlanta Falcons, falcons (26) Carolina Panthers, panthers (27) New Orleans Saints, saints (28) Tampa Bay Buccaneers, buccaneers (29) Arizona Cardinals, cardinals (30) St. Louis Rams, rams (31) San Francisco 49ers, San Francisco 49'ers, 49ers (32) Seattle Seahawks, seahawks

English Dataset: car-makers

(1) Acura (2) Alfa Romeo (3) Aston Martin, Aston-Martin (4) Audi (5) Bentley (6) BMW (7) Buick (8) Cadillac (9) Chevrolet, Chevy (10) Chrysler (11) Citroen, Citroën (12) Daewoo (13) Daihatsu (14) Dodge (15) Ferrari (16) Fiat (17) Ford, Ford Motor Company (18) Geo (19) Honda (20) Hummer (21) Hyundai, Hyundai Motor Company (22) Infiniti (23) Isuzu (24) Jaguar (25) Jeep (26) Kia, Kia Motors (27) Lamborghini (28) Lancia (29) Land Rover, Land-Rover (30) Lexus (31) Lincoln (32) Lotus (33) Maserati (34) Mazda (35) Mercedes-Benz, Mercedes Benz, Mercedes (36) Mercury (37) MINI (38) Mitsubishi (39) Nissan (40) Oldsmobile (41) Opel (42) Peugeot (43) Plymouth (44) Pontiac (45) Porsche (46) Renault (47) Rolls-Royce, Rolls Royce (48) Saab, Saab Automobile (49) Saturn (50) SEAT (51) Smart (52) Subaru (53) Suzuki (54) Toyota (55) Volkswagen, VW (56) Volvo, Volvo Cars

English Dataset: us-presidents

(1) George Washington (2) John Adams (3) Thomas Jefferson (4) James Madison (5) James Monroe (6) John Quincy Adams, John Q. Adams (7) Andrew Jackson (8) Martin Van Buren (9) William Harrison, William H. Harrison, William Henry Harrison (10) John Tyler (11) James Polk, James K. Polk, James Knox Polk (12) Zachary Taylor (13) Millard Fillmore (14) Franklin Pierce (15) James Buchanan (16) Abraham Lincoln (17) Andrew Johnson (18) Ulysses Grant, Ulysses S. Grant, Ulysses Simpson Grant (19) Rutherford Hayes, Rutherford B. Hayes, Rutherford Bitchard Hayes (20) James Garfield, James A. Garfield, James Abram Garfield (21) Chester Arthur, Chester A. Arthur, Chester Alan Arthur (22) Grover Cleveland, Stephen Grover Cleveland (23) Benjamin Harrison (24) William McKinley (25) Theodore Roosevelt, Theodore Roosevelt II (26) William Taft, William H. Taft, William Howard Taft (27) Woodrow Wilson, Thomas Woodrow Wilson (28) Warren Harding, Warren G. Harding, Warren Gamaliel Harding (29) Calvin Coolidge, John Calvin Coolidge, John Calvin Coolidge Jr. (30) Herbert Hoover, Herbert C. Hoover, Herbert Clark Hoover (31) Franklin Roosevelt, Franklin D. Roosevelt, Franklin Delano Roosevelt (32) Harry Truman, Harry S. Truman (33) Dwight Eisenhower, Dwight D. Eisenhower, Dwight David Eisenhower (34) John Kennedy, John F. Kennedy, John Fitzgerald Kennedy (35) Lyndon Johnson, Lyndon B. Johnson, Lyndon Baines Johnson (36) Richard Nixon, Richard M. Nixon, Richard Milhous Nixon (37) Gerald Ford, Gerald R. Ford, Gerald Rudolph Ford, Gerald Rudolph Ford Jr. (38) Jimmy Carter, James E. Carter, James Earl Carter, James Earl Carter Jr. (39) Ronald Reagan, Ronald W. Reagan, Ronald Wilson Reagan (40) George Bush, George H. Bush, George H. W. Bush, George Herbert Walker Bush, George HW Bush, George H.W. Bush (41) Bill Clinton, Bill J. Clinton, Bill Jefferson Clinton, William Clinton, William J. Clinton, William Jefferson Clinton (42) George Walker Bush, George W. Bush, George Bush

English Dataset: us-states

(1) Alaska (2) Alabama (3) Arkansas (4) Arizona (5) California (6) Colorado (7) Connecticut (8) Delaware (9) Florida (10) Georgia (11) Hawaii (12) Iowa (13) Idaho (14) Illinois (15) Indiana (16) Kansas (17) Kentucky (18) Louisiana (19) Massachusetts (20) Maryland (21) Maine (22) Michigan (23) Minnesota (24) Missouri (25) Mississippi (26) Montana (27) North Carolina (28) North Dakota (29) Nebraska (30) New Hampshire (31) New Jersey (32) New Mexico (33) Nevada (34) New York (35) Ohio (36) Oklahoma (37) Oregon (38) Pennsylvania (39) Rhode Island (40) South Carolina (41) South Dakota (42) Tennessee (43) Texas (44) Utah (45) Virginia (46) Vermont (47) Washington (48) Wisconsin (49) West Virginia (50) Wyoming

English Dataset: cmu-buildings

(1) Alumni House (2) Baker Hall (3) Boss House, Boss Hall (4) Bramer House (5) Cathedral Mansions (6) Collaborative Innovation Center (7) College of Fine Arts (8) Cyert Hall (9) Doherty Apartments (10) Doherty Hall (11) Donner House, Donner Hall (12) Facilities Management Services Building (13) Fairfax Apartments (14) Fraternity Quadrangle (15) Skibo Gymnasium, Gymnasium (16) Hamburg Hall (17) Hamerschlag Hall (18) Hamerschlag House (19) Henderson House, Henderson Hall (20) Hunt Library (21) Information Networking Institute (22) London Terrace Apartments (23) Margaret Morrison Apartments (24) Margaret Morrison Carnegie Hall, Margaret Morrison (25) Margaret Morrison Sorority Houses (26) Marybelle Apartments (27) McGill House, McGill Hall (28) Mellon Institute (29) Morewood Gardens (30) Mudge House (31) Neville Apartments (32) New House (33) Newell-Simon Hall (34) Planetary Robotics Building (35) Porter Hall (36) Posner Center (37) Posner Hall, Tepper School of Business (38) Publications/Printing Building, Publications and Printing Building (39) Purnell Center for the Arts, Purnell Center (40) Resnik House (41) Roberts Engineering Hall, Roberts Hall of Engineering, Roberts Hall (42) Roselawn Terrace (43) Saxony Apartments (44) Scaife Hall (45) Scobell House (46) Shady Oak Apartments (47) Shirley Apartments (48) Smith Hall (49) Software Engineering Institute (50) Spirit House (51) Tech House (52) University Center (53) Warner Hall (54) Wean Hall (55) Webster Hall (56) Welch Hall, Welch House (57) West Wing (58) Whitfield Hall (59) Woodlawn Apartments

English Dataset: diseases

(1) Avadhi (2) Acne (3) Adenoma (4) Ageing (5) AIDS, Acquired Immune Deficiency Syndrome (6) Albinism (7) Alcoholic hepatitis (8) Alopecia (9) Alzheimer's disease, Alzheimer (10) Amblyopia (11) Amoebiasis, Amebiasis (12) Anemia (13) Aneurysm (14) Anosmia (15) Anotia (16) Anthrax (17) Appendicitis (18) Apraxia (19) Argyria (20) Arteritis (21) Arthritis (22) Aseptic meningitis (23) Asthenia (24) Asthma (25) Atherosclerosis (26) Athetosis (27) Athlete's foot (28) Atrophy (29) Autism (30) Beriberi (31) Bipolar disorder (32) Bladder infection (33) Botulism (34) Brucellosis (35) Bubonic plague (36) Brain cancer (37) Calculi (38) Campylobacter infection (39) Cancer (40) Candidiasis (41) Cardiac arrest (42) Chagas disease (43) Chalazion (44) Chan-

croid (45) Cherubism (46) Chickenpox (47) Chlamydia (48) Cholera (49) Chordoma (50) Chorea (51) Chronic fatigue syndrome (52) Cleft lip (53) Coccidioidomycosis (54) Cold sore (55) Colitis (56) Color blindness (57) Common cold (58) Condyloma (59) Congestive heart disease (60) Coronary heart disease (61) Cowpox (62) Cretinism (63) Cystic fibrosis (64) Dermatophytosis (65) Diabetes mellitus (66) Diaper rash (67) Diphtheria (68) Ebola (69) E. coli poisoning (70) Encephalitis (71) Foodborne illness (72) Genital warts (73) Gonorrhoea (74) Glandular fever (75) Hemophilia A (76) Hepatitis A (77) Hepatitis B (78) Hepatitis C (79) Hepatitis E (80) Herpes (81) Huntington's disease (82) Hypertension (83) Headache (84) Ichthyosis (85) Influenza (86) Interstitial cystitis (87) Iritis (88) Iron-deficiency anemia (89) Irritable bowel syndrome (90) Jaundice (91) Keloids (92) Keratosis pilaris (93) Kuru (94) Kwashiorkor (95) Lazy eye (96) Lead poisoning (97) Legionellosis (98) Leishmaniasis (99) Leprosy (100) Leptospirosis (101) Listeriosis (102) Leukemia (103) Loiasis (104) Lupus erythematosus (105) Lyme disease (106) Lymphogranuloma venereum (107) Lymphoma (108) Malaria (109) Marburg fever (110) Measles (111) Melioidosis (112) Meniere's disease, Ménière's disease, Meniere, Ménierè (113) Meningitis (114) Meningococcemia (115) Migraine (116) Multiple myeloma (117) Multiple Sclerosis (118) Mumps (119) Muscular dystrophy (120) Myasthenia gravis (121) Myelitis (122) Myoclonus (123) Myopathy (124) Myopia (125) Myxedema (126) Neoplasm (127) Non-gonococcal urethritis (128) Obsessive-compulsive disorder (129) Obesity (130) Osteoarthritis (131) Pancreatitis (132) Paratyphoid fever (133) Parkinson's disease (134) Pelvic inflammatory disease (135) Peritonitis (136) Periodontal disease (137) Pertussis (138) Phenylketonuria (139) Pityriasis rosea (140) Plague, bubonic, septicemic, pneumonic, pharyngeal (141) Pneumonia (142) Polio, Poliomyelitis (143) Porphyria (144) Progeria (145) Prostatitis (146) Psittacosis (147) Psoriasis (148) Pubic lice (149) Q fever (150) Rabies (151) Raynaud's disease, Raynaud (152) Repetitive strain injury, RSI (153) Rheumatic fever (154) Rheumatoid arthritis (155) Rickets (156) Rift Valley fever (157) Rocky Mountain spotted fever (158) Rubella (159) Rheumatic heart disease (160) Salmonella poisoning (161) Salmonellosis (162) Scabies (163) Scarlet fever (164) Sciatica (165) Schizophrenia (166) Scleroderma (167) Scurvy (168) Sepsis (169) SARS, Severe acute respiratory syndrome (170) Shigellosis (171) Shingles (172) Shock (173) Sickle-cell disease (174) Siderosis (175) Silicosis (176) Stevens-Johnson syndrome (177) Strabismus (178) Strep throat (179) Streptococcal infection (180) Synovitis (181) Syphilis (182) Tapeworm infection (183) Tay-Sachs disease (184) Teratoma (185) Tetanus (186) Thalassaemia (187) Thrush (188) Tinnitus (189) Toxic shock syndrome (190) Trichomoniasis (191) Trisomy (192) Tuberculosis (193) Tularemia (194) Tungiasis (195) Typhoid, Typhoid fever (196) Typhus (197) Ulcerative colitis (198) Uremia (199) Urticaria (200) Uveitis (201) Varicella (202) Vasovagal syncope (203) Vitiligo (204) Warkany syndrome (205) Warts (206) Yellow fever (207) Yaws

English Dataset: periodic-comets

(1) Halley (2) Encke (3) Biela (4) Faye (5) Brorsen (6) d'Arrest (7) Pons-Winnecke (8) Tuttle (9) Tempel, Tempel 1 (10) Tempel, Tempel 2 (11) Tempel-Swift-LINEAR (12) Pons-Brooks (13) Olbers (14) Wolf (15) Finlay (16) Brooks, Brooks 2 (17) Holmes (18) Perrine-Mrkos (19) Borrelly (20) Westphal (21) Giacobini-Zinner (22) Kopff (23) Brorsen-Metcalf (24) Schaumasse (25) Neujmin, Neujmin 2 (26) Grigg-Skjellerup (27) Crommelin (28) Neujmin, Neujmin 1 (29) Schwassmann-Wachmann, Schwassmann-Wachmann 1 (30) Reinmuth, Reinmuth 1 (31) Schwassmann-Wachmann, Schwassmann-Wachmann 2 (32) Comas Sola, Comas Solá (33) Daniel (34) Gale (35) Herschel-Rigollet (36) Whipple (37) Forbes (38) Stephan-Oterma (39) Oterma (40) Vaisala, Vaisala 1, Väisälä, Väisälä 1 (41) Tuttle-Giacobini-Kresák, Tuttle-Giacobini-Kresak (42) Neujmin, Neujmin 3 (43) Wolf-Harrington (44) Reinmuth, Reinmuth 2 (45) Honda-Mrkos-Pajdusakova, Honda-Mrkos-Pajdušáková (46) Wirtanen (47) Ashbrook-Jackson (48) Johnson (49) Arend-Rigaux (50) Arend (51) Harrington (52) Harrington-Abell (53) Van Biesbroeck (54) de Vico-Swift-NEAT (55) Tempel-Tuttle (56) Slaughter-Burnham (57) du Toit-Neujmin-Delporte (58) Jackson-Neujmin (59) Kearns-Kwee (60) Tsuchinshan, Tsuchinshan 2 (61) Shajn-Schaldach (62) Tsuchinshan, Tsuchinshan 1 (63) Wild, Wild 1 (64) Swift-Gehrels (65) Gunn (66) du Toit (67) Churyumov-Gerasimenko (68) Klemola (69) Taylor (70) Kojima (71) Clark (72) Denning-Fujikawa (73) Schwassmann-Wachmann, Schwassmann-Wachmann 3 (74) Smirnova-Chernykh (75) Kohoutek (76) West-Kohoutek-Ikemura (77) Longmore (78) Gehrels, Gehrels 2 (79) du Toit-Hartley (80) Peters-Hartley (81) Wild, Wild 2 (82) Gehrels, Gehrels 3 (83) Russell, Russell 1 (84) Giclas (85) Boethin (86) Wild, Wild 3 (87) Bus (88) Howell (89) Russell, Russell 2 (90) Gehrels, Gehrels 1 (91) Russell, Russell 3 (92) Sanguin (93) Lovas, Lovas 1 (94) Russell, Russell 4 (95) Chiron (96) Machholz, Machholz 1 (97) Metcalf-Brewington (98) Takamizawa (99) Kowal, Kowal 1 (100) Hartley, Hartley 1 (101) Chernykh (102) Shoemaker, Shoemaker 1 (103) Hartley, Hartley 2 (104) Kowal, Kowal 2 (105) Singer Brewster (106) Schuster (107) Wilson-Harrington (108) Ciffreo, Ciffréo (109) Swift-Tuttle (110) Hartley, Hartley 3 (111) Helin-Roman-Crockett (112) Urata-Niijima (113) Spitaler (114) Wiseman-Skiff (115) Maury, Maury 1 (116) Wild, Wild 4 (117) Helin-Roman-Alu, Helin-Roman-Alu 1 (118) Shoemaker-Levy, Shoemaker-Levy 4 (119) Parker-Hartley (120) Mueller, Mueller 1 (121) Shoemaker-Holt, Shoemaker-Holt 2 (122) de Vico (123) West-Hartley (124) Mrkos (125) Spacewatch, Spacewatch 1 (126) IRAS (127) Holt-Olmstead (128) Shoemaker-Holt, Shoemaker-Holt 1 (129) Shoemaker-Levy, Shoemaker-Levy 3 (130) McNaught-Hughes (131) Mueller, Mueller 2 (132) Helin-Roman-Alu, Helin-Roman-Alu 2 (133) Elst-Pizarro (134) Kowal-Vavrova, Kowal-Vávrová (135) Shoemaker-Levy, Shoemaker-Levy 8 (136) Mueller, Mueller 3 (137) Shoemaker-Levy, Shoemaker-Levy 2 (138) Shoemaker-Levy, Shoemaker-Levy 7 (139) Vaisala-Oterma, Väisälä-Oterma (140) Bowell-Skiff (141) Machholz, Machholz 2 (142) Ge-Wang (143) Kowal-Mrkos (144) Kushida (145) Shoemaker-Levy, Shoemaker-Levy 5 (146) Shoemaker-LINEAR (147) Kushida-Muramatsu (148) Anderson-LINEAR (149) Mueller, Mueller 4 (150) LONEOS, LONEOS 3 (151) Helin (152) Helin-Lawrence (153) Ikeya-Zhang (154) Brewington (155) Shoemaker, Shoemaker 3 (156) Russell-LINEAR (157) Tritton (158) Kowal-LINEAR (159) LONEOS, LONEOS 7 (160) LINEAR, LINEAR 43 (161) Hartley-IRAS (162) Siding Spring, Siding Spring 2 (163) NEAT, NEAT 21 (164) Christensen, Christensen 2 (165) LINEAR, LINEAR 10 (166) NEAT, NEAT 8 (167) CINEOS (168) Hergenrother, Hergenrother 1 (169) NEAT, NEAT 22 (170) Christensen, Christensen 4 (171) Spahr, Spahr 2 (172) Yeung (173) Mueller, Mueller 5 (174) Echeclus (175) Hergenrother, Hergenrother 2 (176) LINEAR, LINEAR 52 (177) Barnard, Barnard 2 (178) Hug-Bell (179) Jedicke, Jedicke 1 (180) NEAT, NEAT 3 (181) Shoemaker-Levy, Shoemaker-Levy 6 (182) LONEOS, LONEOS 6 (183) Korlevic-Juric, Korlević-Jurić (184) Lovas, Lovas 2 (185) Petriew (186) Garradd

Chinese Dataset: disney-movies

(1) 白雪公主 (2) 木偶奇遇記 (3) 幻想曲 (4) 小飛象 (5) 小鹿斑比 (6) 致候吾友 (7) 三騎士 (8) 爲我譜上樂章 (9) 米奇與魔豆 (10) 旋律時光 (11) 伊老師與小蟾蜍大歷險 (12) 仙履奇緣 (13) 愛麗絲夢遊仙境 (14) 小飛俠 (15) 小姐與流氓 (16) 睡美人 (17) 101 忠狗, １０１忠狗 (18) 石中劍 (19) 森林王子 (20) 貓兒歷險記 (21) 羅賓漢 (22) 小熊維尼歷險記 (23) 救難小英雄 (24) 狐狸與獵狗 (25) 黑神鍋傳奇 (26) 妙妙探 (27) 奧麗華歷險記 (28) 小美人魚 (29) 救難小英雄澳洲歷險記 (30) 美女與野獸 (31) 阿拉丁 (32) 獅子王 (33) 風中奇緣 (34) 鐘樓怪人 (35) 大力士 (36) 花木蘭 (37) 泰山 (38) 幻想曲 2000, 幻想曲２０００ (39) 變身國王 (40) 失落的帝國 (41) 星際寶貝, 星際寶貝史迪奇 (42) 星銀島 (43) 熊的傳說 (44 放牛吃草 (45) 四眼天雞 (46) 未來小子 (47) 美國小狗 (48) 長髮姑娘 (49) 青蛙公主

Chinese Dataset: constellations

(1) 仙女座, 仙女 (2) 唧筒座, 唧筒 (3) 天燕座, 天燕 (4) 寶瓶座, 寶瓶, 宝瓶座, 宝瓶 (5) 天鷹座, 天鷹, 天鹰座, 天鹰 (6) 天壇座, 天壇, 天坛座, 天坛 (7) 白羊座, 白羊 (8) 御夫座, 御夫 (9) 牧夫座, 牧夫 (10) 雕具座, 雕具, 彫具座, 彫具 (11) 鹿豹座, 鹿豹 (12) 巨蟹座, 巨蟹 (13) 獵犬座, 獵犬, 猎犬座, 猎犬 (14) 大犬座, 大犬 (15) 小犬座, 小犬 (16) 摩羯座, 摩羯, 魔羯座, 魔羯 (17) 船底座, 船底 (18) 仙后座, 仙后 (19) 半人馬座, 半人馬, 半人马座, 半人马 (20) 仙王座, 仙王 (21) 鯨魚座, 鯨魚, 鲸鱼座, 鲸鱼 (22) 蝘蜓座, 蝘蜓座, 堰蜓座, 堰蜓座, 偃蜓座, 偃蜓座, 蝘蜓, 蝘蜓, 堰蜓, 堰蜓, 偃蜓, 偃蜓 (23) 圓規座, 圓規, 圆规座, 圆规 (24) 天鴿座, 天鴿, 天鸽座, 天鸽 (25) 后髮座, 后髮, 后发座, 后发 (26) 南冕座, 南冕 (27) 北冕座, 北冕 (28) 烏鴉座, 烏鴉, 乌鸦座, 乌鸦 (29) 巨爵座, 巨爵 (30) 南十字座, 南十字 (31) 天鵝座, 天鵝, 天鹅座, 天鹅 (32) 海豚座, 海豚 (33) 劍魚座, 劍魚, 剑鱼座, 剑鱼 (34) 天龍座, 天龍, 天龙座, 天龙 (35) 小馬座, 小馬, 小马座, 小马 (36) 波江座, 波江 (37) 天爐座, 天爐, 天炉座, 天炉 (38) 雙子座, 雙子, 双子座, 双子 (39) 天鶴座, 天鶴, 天鹤座, 天鹤 (40) 武仙座, 武仙 (41) 時鐘座, 時鐘, 时钟座, 时钟 (42) 長蛇座, 長蛇, 长蛇座, 长蛇 (43) 水蛇座, 水蛇 (44) 印第安座, 印第安, 印地安座, 印地安 (45) 蠍虎座, 蠍虎, 蝎虎座, 蝎虎 (46) 獅子座, 獅子, 狮子座, 狮子 (47) 小獅座, 小獅 小狮座, 小狮 (48) 天兔座, 天兔 (49) 天秤座, 天秤 (50) 豺狼座, 豺狼, 天狼座, 天狼 (51) 天貓座, 天貓, 天猫座, 天猫 (52) 天琴座, 天琴 (53) 山案座, 山案 (54) 顯微鏡座, 顯微鏡, 显微镜座, 显微镜 (55) 麒麟座, 麒麟 (56) 蒼蠅座, 蒼蠅, 苍蝇座, 苍蝇 (57) 矩尺座, 矩尺 (58) 南極座, 南極, 南极座, 南极 (59) 蛇夫座, 蛇夫 (60) 獵戶座, 獵戶, 猎户座, 猎户 (61) 孔雀座, 孔雀 (62) 飛馬座, 飛馬, 飞马座, 飞马 (63) 英仙座, 英仙 (64) 鳳凰座, 鳳凰, 凤凰座, 凤凰 (65) 繪架座, 繪架, 绘架座, 绘架 (66) 雙魚座, 雙魚, 双鱼座, 双鱼 (67) 南魚座, 南魚, 南鱼座, 南鱼 (68) 船尾座, 船尾 (69) 羅盤座, 羅盤, 罗盘座, 罗盘 (70) 網罟座, 網罟, 网罟座, 网罟 (71) 天箭座, 天箭 (72) 人馬座, 人馬, 人马座, 人马 (73) 天蠍座, 天蠍, 天蝎座, 天蝎 (74) 玉夫座, 玉夫 (75) 盾牌座, 盾牌 (76) 巨蛇座, 巨蛇 (77) 六分儀座, 六分儀, 六分仪座, 六分仪 (78) 金牛座, 金牛 (79) 望遠鏡座, 望遠鏡, 望远镜座, 望远镜 (80) 三角座, 三角 (81) 南三角座, 南三角 (82) 杜鵑座, 杜鵑, 杜鹃座, 杜鹃 (83) 大熊座, 大熊 (84) 小熊座, 小熊 (85) 船帆座, 船帆 (86) 室女座, 室女, 處女座, 處女 (87) 飛魚座, 飛魚, 飞鱼座, 飞鱼 (88) 狐狸座, 狐狸

Chinese Dataset: countries

(1) 阿富汗, 阿富汗伊斯蘭共和國 (2) 阿爾巴尼亞, 阿爾巴尼亞共和國 (3) 阿爾及利亞, 阿爾及利亞人民民主共和國 (4) 安道爾, 安道爾公國 (5) 安哥拉, 安哥拉共和國 (6) 安地卡及巴布達, 安地卡和巴布達, 安地卡, 安地卡巴布達 (7) 阿根廷, 阿根廷共和國 (8) 亞美尼亞, 亞美尼亞共和國 (9) 澳洲, 澳大利亞, 澳大利亞聯邦 (10) 奧地利, 奧地利共和國 (11) 亞塞拜然, 亞塞拜然共和國 (12) 巴哈馬, 巴哈馬聯邦 (13) 巴林, 巴林王國 (14) 孟加拉, 孟加拉人民共和國 (15) 巴貝多 (16) 白俄羅斯, 白俄羅斯共和國 (17) 比利時, 比利時王國 (18) 貝里斯 (19) 貝南, 貝南共和國 (20) 不丹, 不丹王國 (21) 玻利維亞, 玻利維亞共和國 (22) 波士尼亞, 波士尼亞赫塞哥維納 (23) 波札那, 波札那共和國 (24) 巴西, 巴西聯邦共和國 (25) 汶萊, 汶萊達魯薩蘭國 (26) 保加利亞, 保加利亞共和國 (27) 布吉納法索 (28) 蒲隆地, 蒲隆地共和國 (29) 柬埔寨, 柬埔寨王國 (30) 喀麥隆, 喀麥隆共和國 (31) 加拿大 (32) 維德角, 維德角共和國 (33) 中非, 中非共和國 (34) 查德, 查德共和國 (35) 智利, 智利共和國 (36) 中國, 中國大陸, 中華人民共和國 (37) 哥倫比亞, 哥倫比亞共和國 (38) 葛摩, 葛摩聯盟 (39) 剛果, 剛果民主共和國 (40) 剛果, 剛果共和國 (41) 哥斯大黎加, 哥斯大黎加共和國 (42) 象牙海岸, 象牙海岸共和國 (43) 克羅埃西亞, 克羅埃西亞共和國 (44) 古巴, 古巴共和國 (45) 賽普勒斯, 塞普勒斯, 賽普勒斯共和國, 塞普勒斯共和國 (46) 捷克, 捷克共和國 (47) 丹麥, 丹麥王國 (48) 吉布地, 吉布地共和國 (49) 多米尼克, 多米尼克國 (50) 多明尼加, 多明尼加共和國 (51) 厄瓜多, 厄瓜多共和國 (52) 東帝汶, 東帝汶民主共和國 (53) 埃及, 埃及阿拉伯共和國 (54) 薩爾瓦多, 薩爾瓦多共和國 (55) 赤道幾內亞, 赤道幾內亞共和國 (56) 厄利垂亞, 厄利垂亞國 (57) 愛沙尼亞, 愛沙尼亞共和國 (58) 衣索比亞, 衣索比亞聯邦民主共和國 (59) 斐濟, 斐濟群島共和國 (60) 芬蘭, 芬蘭共和國 (61) 法國, 法蘭西共和國 (62) 加彭, 加彭共和國 (63) 甘比亞, 甘比亞共和國 (64) 喬治亞 (65) 德國, 德意志聯邦共和國 (66) 迦納, 迦納共和國 (67) 希臘, 希臘共和國 (68) 格瑞那達 (69) 瓜地馬拉, 瓜地馬拉共和國 (70) 幾內亞, 幾內亞共和國 (71) 幾內亞比索, 幾內亞比索共和國 (72) 蓋亞那, 圭亞那, 蓋亞那合作共和國, 圭亞那合作共和國 (73) 海地, 海地共和國 (74) 宏都拉斯, 宏都拉斯共和國 (75) 匈牙利, 匈牙利共和國 (76) 冰島, 冰島共和國 (77) 印度, 印度共和國 (78) 印尼, 印度尼西亞, 印度尼西亞共和國 (79) 伊朗, 伊朗伊斯蘭共和國 (80) 伊拉克, 伊拉克共和國 (81) 愛爾蘭 (82) 以色列, 以色列國 (83) 義大利, 義大利共和國, 意大利 (84) 牙買加, 牙買加共和國 (85) 日本, 日本國 (86) 約旦, 約旦哈希姆王國 (87) 哈薩克, 哈薩克共和國 (88) 肯亞, 肯亞共和國 (89) 吉里巴斯, 吉里巴斯共和國 (90) 朝鮮, 朝鮮民主主義人民共和國, 北韓 (91) 韓國, 大韓民國, 南韓 (92) 科威特, 科威特國 (93) 吉爾吉斯, 吉爾吉斯共和國 (94) 寮國, 寮人民民主共和國 (95) 拉脫維亞, 拉托維亞, 拉脫維亞共和國, 拉托維亞共和國 (96) 黎巴嫩, 黎巴嫩共和國 (97) 賴索托, 賴索托王國 (98) 賴比瑞亞, 賴比瑞亞共和國 (99) 利比亞, 大利比亞阿拉伯人民社會主義群眾國 (100) 列支敦斯登, 列支敦斯登大公國 (101) 立陶宛, 立陶宛共和國 (102) 盧森堡, 盧森堡大公國 (103) 馬其頓, 馬其頓共和國 (104) 馬達加斯加, 馬達加斯加共和國 (105) 馬拉威, 馬拉威共和國 (106) 馬來西亞 (107) 馬爾地夫, 馬爾地夫共和國 (108) 馬利, 馬利共和國 (109) 馬爾他, 馬爾他共和國 (110) 馬紹爾群島, 馬紹爾群島共和國 (111) 茅利塔尼亞, 茅利塔尼亞伊斯蘭共和國 (112) 模里西斯, 模里西斯共和國 (113) 墨西哥, 墨西哥合眾國 (114) 密克羅尼西亞, 密克羅尼西亞聯邦 (115) 摩爾多瓦, 摩爾多瓦共和國 (116) 摩納哥, 摩納哥大公國 (117) 蒙古, 蒙古國 (118) 蒙特內哥羅, 蒙特內哥羅共和國 (119) 摩洛哥, 摩洛哥王國 (120) 莫三比克, 莫三比克共和國 (121) 緬甸, 緬甸聯邦 (122) 奈米比亞, 納米比亞, 奈米比亞共和國, 納米比亞共和國 (123) 諾魯, 諾魯共和國 (124) 尼泊爾, 尼泊爾王國 (125) 荷蘭, 荷蘭王國 (126) 紐西蘭, 新西蘭 (127) 尼加拉瓜, 尼加拉瓜共和國 (128) 尼日, 尼日共和國 (129) 奈及利亞, 奈及利亞聯邦共和國 (130) 挪威, 挪威王國 (131) 阿曼, 阿曼蘇丹國 (132) 巴基斯坦, 巴基斯坦伊斯蘭共和國 (133) 帛琉, 帛琉共和國 (134) 巴勒斯坦, 巴勒斯坦國 (135) 巴拿馬, 巴拿馬共和國 (136) 巴布亞紐幾內亞, 巴布亞新幾內亞, 巴布亞紐幾內亞獨立國, 巴布亞新幾內亞獨立國 (137) 巴拉圭, 巴拉圭共和國 (138) 秘魯, 祕魯, 秘魯共和國 (139) 菲律賓, 菲律賓共和國 (140) 波蘭, 波蘭共和國 (141) 葡萄牙, 葡萄牙共和國 (142) 卡達, 卡達酋長國 (143) 羅馬尼亞 (144) 俄羅斯, 俄羅斯聯邦 (145) 盧安達, 盧安達共和國 (146) 聖克里斯多福及尼維斯, 聖克里斯多福, 聖克里斯多福及尼維斯聯邦 (147) 聖露西亞 (148) 聖文森及格瑞那丁, 聖文森 (149) 薩摩亞, 薩摩亞獨立國 (150) 聖馬利諾, 聖馬利諾共和國 (151) 聖多美普林西比, 聖多美和普林西比, 聖多美及普林西比, 聖多美和普林西比民主共和國 (152) 沙烏地阿拉伯, 沙烏地阿拉伯王國 (153) 塞內加爾, 塞內加爾共和國 (154) 塞爾維亞 (155) 塞席爾, 塞席爾共和國 (156) 獅子山, 獅子山共和國 (157) 新加坡, 新加坡共和國 (158) 斯洛

伐克, 斯洛伐克共和國 (159) 斯洛維尼亞, 斯洛維尼亞共和國 (160) 索羅門群島 (161) 索馬利亞 (162) 南非, 南非共和國 (163) 西班牙, 西班牙王國 (164) 斯里蘭卡, 斯里蘭卡民主社會主義共和國 (165) 蘇丹, 蘇丹共和國 (166) 蘇利南, 蘇利南共和國 (167) 史瓦濟蘭, 史瓦濟蘭王國 (168 瑞典, 瑞典王國 (169) 瑞士, 瑞士邦聯 (170) 敘利亞, 阿拉伯敘利亞共和國 (171) 台灣, 臺灣, 中華民國 (172) 塔吉克, 塔吉克共和國 (173) 坦尚尼亞, 坦尚尼亞聯合共和國 (174) 泰國, 泰王國 (175) 多哥, 多哥共和國 (176) 東加, 東加王國 (177) 千里達及托巴哥, 千里達, 特立尼達和多巴哥, 千里達及托巴哥共和國, 特立尼達和多巴哥共和國, 千里達托巴哥 (178) 突尼西亞, 突尼斯, 突尼西亞共和國, 突尼斯共和國 (179) 土耳其, 土耳其共和國 (180) 土庫曼 (181) 吐瓦魯 (182) 烏干達, 烏干達共和國 (183) 烏克蘭 (184) 阿聯, 阿拉伯, 阿拉伯聯合大公國 (185) 英國, 大不列顛及北愛爾蘭聯合王國 (186) 美國, 美利堅合眾國 (187) 烏拉圭, 烏拉圭東岸共和國 (188) 烏茲別克, 烏茲別克斯坦, 烏茲別克共和國 (189) 萬那杜, 萬那杜共和國 (190) 教廷, 梵蒂岡, 梵蒂岡城國 (191) 委內瑞拉, 委內瑞拉玻利瓦爾共和國 (192) 越南, 越南社會主義共和國 (193) 西撒哈拉 (194) 葉門, 葉門共和國 (195) 尚比亞, 尚比亞共和國 (196) 辛巴威, 辛巴威共和國

Chinese Dataset: mlb-teams

(1) 亞特蘭大勇士, 勇士 (2) 佛羅里達馬林魚, 馬林魚 (3) 紐約大都會, 大都會 (4) 費城費城人, 費城人 (5) 華盛頓國民, 國民 (6) 芝加哥小熊, 小熊 (7) 辛辛那提紅人, 辛辛那堤紅人, 紅人 (8) 休士頓太空人, 太空人 (9) 密爾瓦基釀酒人, 釀酒人 (10) 匹茲堡海盜, 匹茨堡海盜, 海盜 (11) 聖路易紅雀, 聖路易斯紅雀, 紅雀 (12) 亞利桑那響尾蛇, 亞歷桑那響尾蛇, 亞歷桑納響尾蛇, 亞利桑納響尾蛇, 響尾蛇 (13) 科羅拉多洛磯, 科羅拉多落磯, 洛磯, 落磯 (14) 洛杉磯道奇, 布魯克林道奇, 道奇 (15) 聖地牙哥教士, 聖地亞哥教士, 教士 (16) 舊金山巨人, 巨人 (17) 巴爾的摩金鶯, 巴尔的摩金莺, 金鶯, 金莺 (18) 波士頓紅襪, 紅襪 (19) 紐約洋基, 洋基 (20) 坦帕灣光芒, 坦帕灣魔鬼魚, 光芒, 魔鬼魚 (21) 多倫多藍鳥, 藍鳥 (22) 芝加哥白襪, 白襪 (23) 克里夫蘭印地安人, 克里夫蘭印第安人, 克利夫蘭印地安人, 印地安人, 印第安人 (24) 底特律老虎, 老虎 (25) 堪薩斯皇家, 堪薩斯市皇家, 堪薩斯城皇家, 坎薩斯皇家, 皇家 (26) 明尼蘇達雙城, 雙城 (27) 洛杉磯安那罕天使, 洛杉磯天使, 安那罕天使, 安納罕天使, 安納漢天使, 安那漢天使, 天使 (28) 奧克蘭運動家, 運動家 (29) 西雅圖水手, 水手 (30) 德州遊騎兵, 遊騎兵

Chinese Dataset: nba-teams

(1) 波士頓塞爾提克, 波士頓賽爾提克, 波士頓塞爾蒂克, 波士頓凱爾特人, 塞爾提克, 賽爾提克, 塞爾蒂克, 凱爾特人 (2) 紐澤西籃網, 新澤西籃網, 籃網 (3) 紐約尼克, 紐約尼客, 紐約尼克斯, 尼克, 尼客, 尼克斯 (4) 費城 76 人, 費城 76 人, 費城七十六人, 費城七六人, 76 人, 七十六人, 七六人 (5) 多倫多暴龍, 暴龍, 猛龍 (6) 芝加哥公牛, 公牛 (7) 克里夫蘭騎士, 克裡夫蘭騎士, 克利夫蘭騎士, 騎士 (8) 底特律活塞, 活塞 (9) 印第安那溜馬, 印地安那溜馬, 印地安納溜馬, 印第安納溜馬, 溜馬, 步行者 (10) 密爾瓦基公鹿, 公鹿, 雄鹿 (11) 亞特蘭大老鷹, 老鷹 (12) 夏洛特山貓, 夏洛特黃蜂, 山貓 (13) 邁阿密熱火, 熱火 (14) 奧蘭多魔術, 魔術 (15) 華盛頓巫師, 華盛頓奇才, 巫師, 奇才 (16) 達拉斯小牛, 小牛 (17) 休士頓火箭, 休斯頓火箭, 火箭 (18) 孟斐斯灰熊, 孟菲斯灰熊, 曼菲斯灰熊, 曼斐斯灰熊, 溫哥華灰熊, 灰熊 (19) 紐奧良黃蜂, 夏洛特黃蜂, 黃蜂 (20) 聖安東尼奧馬刺, 聖安東尼市馬刺, 聖安東尼馬刺, 馬刺 (21) 丹佛金塊, 丹佛掘金, 金塊, 掘金 (22) 明尼蘇達灰狼, 明尼蘇達木狼, 明尼蘇達森林狼, 灰狼, 森林狼 (23) 波特蘭拓荒者, 拓荒者, 開拓者 (24) 奧克拉荷馬雷霆, 雷霆, 西雅圖超音速, 超音速 (25) 猶他爵士, 爵士 (26) 金州勇士, 勇士 (27) 洛杉磯快艇, 快艇, 快船 (28) 洛杉磯湖人, 湖人 (29) 鳳凰城太陽, 太陽 (30) 沙加緬度國王, 國王

Chinese Dataset: nfl-teams

(1) 水牛城比爾, 水牛城比尔, 比爾, 比尔 (2) 邁阿密海豚, 迈阿密海豚, 海豚 (3) 新英格蘭愛國者, 新英格兰爱国者, 愛國者, 爱国者 (4) 紐約噴射機, 紐約噴汽機, 紐約噴氣機, 纽约喷汽机, 噴射機, 噴汽機, 噴氣機, 喷汽机 (5) 巴爾的摩烏鴉, 巴尔的摩乌鸦, 摩烏鴉, 摩乌鸦 (6) 辛辛那提孟加拉虎, 辛辛納提孟加拉虎, 辛辛那提孟拉佳虎, 孟加拉虎, 孟拉佳虎 (7) 克里夫蘭布朗, 克裡夫蘭布朗, 克利夫蘭布朗, 克里夫兰布朗, 布朗 (8) 匹茲堡鋼人, 匹茲堡鋼鐵人, 匹兹堡钢人, 鋼人, 鋼鐵人, 钢人 (9) 休斯頓德克薩斯人, 休斯顿德克萨斯人, 休士頓德州人, 德克薩斯人, 德克薩斯人, 德州人 (10) 印第安那小馬, 印城小馬, 印地安那波利斯小馬, 印第安納波利斯小馬, 印第安那小马, 小馬 (11) 傑克遜威爾美洲虎, 傑克森維爾美洲虎, 傑克遜維爾美洲虎, 杰克逊威尔美洲虎, 美洲虎 (12) 田納西泰坦, 田納西巨神, 田纳西巨神, 泰坦, 巨神 (13) 丹佛野馬, 丹佛野马, 野馬, 野马 (14) 堪薩斯城酋長, 堪薩斯酋長, 坎薩斯酋長, 堪萨斯城酋长, 酋長, 酋长 (15) 奧克蘭襲擊者, 奧克蘭突擊者, 奧克蘭突襲者, 奥克兰袭击者, 奧克蘭侵略者, 襲擊者, 突擊者, 袭击者, 侵略者 (16) 聖地牙哥電光, 聖地亞哥電光, 圣地牙哥电光, 電光, 电光 (17) 達拉斯牛仔, 达拉斯牛仔, 牛仔 (18) 紐約巨人, 纽约巨人, 巨人 (19) 費城老鷹, 费城老鹰, 老鷹 (20) 華盛頓紅人, 華盛頓紅皮膚, 華盛頓紅皮, 华盛顿红皮肤, 紅人, 紅皮膚, 红皮肤 (21) 芝加哥熊, 熊 (22) 底特律雄獅, 底特律雄狮, 雄獅 (23) 綠灣包裝工, 綠灣包裝人, 綠灣包裝者, 绿湾包装工, 包裝工, 包裝人, 包裝者 (24) 明尼蘇達維京人, 明尼苏达维京人, 維京人, 维京人 (25) 亞特蘭大獵鷹, 獵鷹 (26) 卡羅萊納黑豹, 卡羅來那黑豹, 卡羅萊那黑豹, 卡羅來納黑豹, 卡羅萊納美洲豹, 卡罗来那黑豹, 黑豹, 美洲豹, 黑豹 (27) 紐奧良聖徒, 紐奧倫聖徒, 紐奧爾良聖徒, 新奧爾良聖徒, 新奥尔良圣徒, 聖徒, 圣徒 (28) 坦帕灣海盜, 坦帕湾海盗, 海盜 (29) 亞利桑那紅雀, 亚利桑那红雀, 亞歷桑納紅雀, 亞歷桑那紅雀, 紅雀 (30) 聖路易公羊, 聖路易斯公羊, 圣路易斯公羊, 公羊 (31) 舊金山 49 人, 舊金山四九人, 舊金山四十九人, 旧金山四十九人, 三藩市 49 人, 三藩市四九人, 三藩市四十九人, 49 人, 四九人, 四十九人 (32) 西雅圖海鷹, 西雅图海鹰, 海鷹

Chinese Dataset: car-makers

(1) 極品, 讴歌, 謳歌 (2) 愛快羅密歐, 愛發羅密歐, 愛快羅蜜歐, 阿尔法·罗密欧, 愛發 (3) 奧斯頓馬丁, 阿斯顿马丁, 阿斯頓馬丁, 雅士頓馬田, 亞斯頓馬汀 (4) 奧迪, 奥迪 (5) 賓特利, 班特利, 本特利, 賓利, 宾利 (6) 寶馬, 宝马 (7) 別克, 標域 (8) 凱迪拉克, 凱迪拉克, 佳特力 (9) 雪佛蘭, 雪佛兰, 雪弗莱, 雪弗萊 (10) 克萊斯勒, 克莱斯勒, 克雷斯勒, 佳士拿 (11) 雪鐵龍, 雪铁龙, 先進, 雪朗 (12) 大宇 (13) 大發, 大发 (14) 道奇, 道濟 (15) 法拉利 (16) 飛雅特, 菲亚特, 菲亞特, 快意 (17) 福特 (18) 吉優 (19) 本田 (20) 悍馬, 悍马 (21) 現代, 现代, 現代汽車 (22) 無限, 无限, 極致, 英菲尼迪 (23) 五十鈴, 五十铃 (24) 捷豹, 積架 (25) 吉普 (26) 起亞, 起亚, 起亞汽車 (27) 林寶堅尼, 兰博基尼, 藍寶堅尼 (28) 蘭吉雅, 兰旗亚, 藍吉亞, 領先 (29) 荒原路華, 路虎, 越野路華, 越野路虎, 路華 (30) 凌志, 雷克萨斯, 樂薩士 (31) 林肯 (32) 蓮花, 莲花 (33) 瑪莎拉蒂, 玛莎拉蒂, 瑪莎雷蒂 (34) 馬自達, 马自达, 萬事得 (35) 賓士, 朋馳, 奔驰, 平治 (36) 水星 (37) 迷你 (38) 三菱 (39) 日產, 日产, 尼桑 (40) 奧斯摩比, 奥兹莫比尔, 奧士無比 (41) 歐寶, 欧宝, 歐普 (42) 寶獅, 标致, 標致, 標緻 (43) 順風, 普利茅斯 (44) 龐帝克, 庞蒂亚克, 龐迪亞克, 龐蒂雅克, 潘迪 (45) 保時捷, 保时捷 (46) 雷諾, 雷诺, 金威龍 (47) 勞斯萊斯, 劳斯莱斯 (48) 紳寶, 萨博, 绅宝 (49) 釷星, 土星 (50) 喜悅, 西亚特, 西亞特, 西雅 (51) 司麥特, 司麥特國際, 精灵 (52) 速霸陸, 斯巴鲁, 斯巴魯, 速霸路, 富士 (53) 鈴木, 铃木 (54) 豐田, 丰田 (55) 福斯, 大众, 大眾, 福士 (56) 富豪, 沃尔沃, 沃爾沃

Chinese Dataset: us-presidents

(1) 華盛頓, 华盛顿, 喬治·華盛頓 (2) 亞當斯, 亚当斯, 約翰·亞當斯 (3) 傑弗遜, 傑佛遜, 托馬斯·傑弗遜 (4) 麥迪遜, 詹姆斯·麥迪遜 (5) 門羅, 詹姆斯·門羅 (6) 亞當斯, 亚当斯, 約翰·昆西·亞當斯 (7) 傑克遜, 杰克逊, 安德魯·傑克遜 (8) 范布倫, 範布倫, 范布伦, 馬丁·范布倫 (9) 哈里森, 哈里遜哈森, 亨利·哈里森, 威廉·亨利·哈里森 (10) 泰勒, 約翰·泰勒 (11) 波爾克, 波克, 詹姆斯·諾克斯·波爾克 (12) 泰勒, 札克利·泰勒, 扎卡裡·泰勒 (13) 菲爾莫爾, 菲尔莫尔, 費爾摩, 米勒德·菲爾莫爾 (14) 皮爾斯, 皮尔斯, 富蘭克林·皮爾斯 (15) 布坎南, 詹姆斯·布坎南 (16) 林肯, 里根, 亞伯拉罕·林肯 (17) 詹森, 安德魯·詹森, 約翰遜, 约翰逊, 強森, 強生 (18) 格蘭特, 葛蘭特, 格兰特, 尤里西斯·格蘭特 (19) 海斯, 拉瑟福德·伯查德·海斯 (20) 加菲爾德, 加菲尔德, 詹姆斯·加菲爾德 (21) 阿瑟, 亞瑟, 艾倫亞瑟, 切斯特·阿瑟, 切斯特·A·阿瑟 (22) 克利夫蘭, 克里夫蘭, 克里夫兰, 克利夫兰, 格羅弗·克利夫蘭 (23) 哈里森, 哈里遜, 班傑明·哈里森, 班哲明·哈里遜 (24) 麥金萊, 麦金莱, 威廉·麥金萊 (25) 羅斯福, 罗斯福, 老羅斯福, 西奧多·羅斯福 (26) 塔夫脫, 塔夫特, 塔虎特, 威廉·霍華德·塔夫脫 (27) 威爾遜, 伍德羅·威爾遜 (28) 哈定, 沃倫·蓋瑪利爾·哈定 (29) 柯立芝, 卡爾文·柯立芝 (30) 胡佛, 赫伯特·胡佛 (31) 羅斯福, 罗斯福, 小羅斯福, 富蘭克林·德拉諾·羅斯福 (32) 杜魯門, 杜鲁门, 哈利·杜魯門, 哈利·S·杜魯門 (33) 艾森豪, 艾森豪威爾, 艾森豪威尔, 德懷特·艾森豪 (34) 甘迺迪, 甘廼迪, 甘乃迪, 肯尼迪, 約翰·甘迺迪 (35) 詹森, 約翰遜, 约翰逊, 強森, 林登·詹森 (36) 尼克森, 尼克遜, 尼克松, 理察·尼克森 (37) 福特, 傑拉爾德·福特 (38) 卡特, 吉米·卡特 (39) 雷根, 列根, 隆納·雷根 (40) 布希, 布什, 布殊, 老布希, 喬治·赫伯特·沃克·布希 (41) 柯林頓, 克林頓, 克林顿, 比爾·柯林頓 (42) 布希, 布什, 布殊, 小布希, 喬治·沃克·布希

Chinese Dataset: us-states

(1) 阿拉斯加州, 阿拉斯加 (2) 阿拉巴馬州, 阿拉巴馬 (3) 阿肯色州, 阿肯色 (4) 亞利桑那州, 亞利桑那 (5) 加州, 加利福尼亞州, 加利福尼亞 (6) 科羅拉多州, 科羅拉多 (7) 康乃狄克州, 康乃迪克州, 康乃狄克, 康乃迪克, 康涅狄格 (8) 德拉瓦州, 戴拉維爾州, 德拉威州, 德拉瓦, 戴拉維爾, 德拉威 (9) 佛羅里達州, 佛羅里達 (10) 喬治亞州, 喬治亞 (11) 夏威夷州, 夏威夷 (12) 愛荷華州, 愛荷華 (13) 愛達荷州, 愛德荷州, 愛達荷, 愛德荷 (14) 伊利諾州, 伊利諾 (15) 印第安納州, 印第安那州, 印地安那州, 印地安納州, 印第安納, 印第安那, 印地安那, 印地安納 (16) 堪薩斯州, 肯薩斯州, 堪薩斯, 肯薩斯 (17) 肯塔基州, 肯塔基 (18) 路易斯安那州, 路易西安納州, 路易西安那州, 路易斯安那, 路易西安納, 路易西安那 (19) 麻州, 麻薩諸塞州, 麻塞諸塞州, 麻塞諸瑟州, 麻薩諸塞, 麻塞諸塞, 麻塞諸瑟 (20) 馬里蘭州, 馬里蘭 (21) 緬因州, 緬因 (22) 密西根州, 密歇根州密西根, 密歇根 (23) 明尼蘇達州, 明尼蘇達 (24) 密蘇里州, 密蘇里 (25) 密西西比州, 密西西比 (26) 蒙大拿州, 蒙大拿 (27) 北卡羅萊納州, 北卡羅來納州, 北卡羅萊那州, 北卡州, 北卡羅萊納, 北卡羅來納, 北卡羅萊那, 北卡 (28) 北達科塔州, 北達科他州, 北達科塔, 北達科他 (29) 內布拉斯加州, 內布拉斯加 (30) 新罕布夏州, 新罕布什州, 新罕布什爾州, 新罕布夏, 新罕布什, 新罕布什爾 (31) 紐澤西州, 新澤西州, 紐澤西, 新澤西 (32) 新墨西哥州, 新墨西哥 (33) 內華達州, 內華達 (34) 紐約州, 紐約 (35) 俄亥俄州, 俄亥俄 (36) 奧克拉荷馬州, 俄克拉荷馬州, 奧克拉荷馬, 俄克拉荷馬 (37) 奧勒崗州, 奧勒岡州, 俄勒岡州, 奧勒崗, 奧勒岡, 俄勒岡 (38) 賓州, 賓夕法尼亞州, 賓夕凡尼亞州, 賓夕尼亞州, 賓夕法尼亞, 賓夕凡尼亞, 賓夕尼亞 (39) 羅德島州, 羅德島 (40) 南卡羅萊納州, 南卡羅來納州, 南卡羅萊那州, 南卡州, 南卡羅萊納, 南卡羅來納, 南卡羅萊那, 南卡 (41) 南達科塔州, 南達科他州, 南達科塔, 南達科他 (42) 田納西州, 田那西州, 田納西, 田那西 (43) 德州, 德克薩斯州, 德克薩斯 (44) 猶他州, 猶他 (45) 維吉尼亞州, 維吉尼亞, 弗吉尼亞 (46) 佛蒙特州, 佛蒙特 (47) 華盛頓州, 華盛頓 (48) 威斯康辛州, 威斯康辛 (49) 西維吉尼亞州, 西維琴尼亞州, 西維吉尼州, 西維吉尼亞, 西維琴尼亞, 西維吉尼 (50) 懷俄明州, 懷俄明

Chinese Dataset: china-dynasties

(1) 夏朝, 夏 (2) 商朝, 商 (3) 周朝, 周 (4) 西周 (5) 東周 (6) 春秋 (7) 戰國 (8) 秦朝, 秦 (9) 西楚 (10) 漢朝, 漢 (11) 西漢 (12) 新朝 (13) 東漢 (14) 三國 (15) 曹魏 (16) 蜀漢 (17) 孫吳 (18) 晉朝, 晉 (19) 西晉 (20) 東晉 (21) 十六國 (22) 漢前趙 (23) 成漢 (24) 前涼 (25) 後趙 (26) 前燕 (27) 前秦 (28) 後秦 (29) 後燕 (30) 西秦 (31) 後涼 (32) 南涼 (33) 南燕 (34) 西涼 (35) 夏 (36) 北燕 (37) 北涼 (38) 南北朝 (39) 魏晉南北朝 (40) 南朝 (41) 南朝宋 (42) 南朝齊 (43) 南朝梁 (44) 南朝陳 (45) 北朝 (46) 北魏 (47) 東魏 (48) 北齊 (49) 西魏 (50) 北周 (51) 隋朝, 隋 (52) 唐朝, 唐 (53) 武周 (54) 五代 (55) 後梁 (56) 後唐 (57) 後晉 (58) 後漢 (59) 後周 (60) 十國 (61) 吳越 (62) 閩國 (63) 南平 (64) 楚國 (65) 吳國 (66) 南唐 (67) 南漢 (68) 北漢 (69) 前蜀 (70) 後蜀 (71) 宋朝, 宋 (72) 北宋 (73) 南宋 (74) 遼朝, 遼 (75) 西遼 (76) 西夏 (77) 金朝, 金 (78) 元朝, 元 (79) 明朝, 明 (80) 清朝, 清 (81) 中華民國, 民國 (82) 中華人民共和國

Chinese Dataset: china-provinces

(1) 河北省, 河北 (2) 河南省, 河南 (3) 山西省, 山西 (4) 山東省, 山东省, 山東, 山东 (5) 遼寧省, 辽宁省, 遼寧, 辽宁 (6) 吉林省, 吉林 (7) 黑龍江省黑龙江省, 黑龍江, 黑龙江 (8) 江蘇省, 江苏省, 江蘇, 江苏 (9) 浙江省, 浙江 (10) 安徽省, 安徽 (11) 福建省, 福建 (12) 江西省, 江西 (13) 湖北省, 湖北 (14) 湖南省, 湖南 (15) 廣東省, 广东省, 廣東, 广东 (16) 海南省, 海南 (17) 四川省, 四川 (18) 貴州省, 贵州省, 貴州, 贵州 (19) 雲南省, 云南省, 雲南, 云南 (20) 陝西省, 陕西省, 陜西省, 陝西, 陕西, 陜西 (21) 甘肅省, 甘肃省, 甘肅, 甘肃 (22) 青海省, 青海

Chinese Dataset: taiwan-cities

(1) 基隆市 (2) 台北市, 臺北市 (3) 台北縣, 臺北縣 (4) 桃園縣 (5) 新竹市 (6) 新竹縣 (7) 苗栗縣 (8) 台中市, 臺中市 (9) 台中縣, 臺中縣 (10) 彰化縣 (11) 南投縣 (12) 雲林縣 (13) 嘉義市 (14) 嘉義縣 (15) 台南市, 臺南市 (16) 台南縣, 臺南縣 (17) 高雄市 (18) 高雄縣 (19) 屏東縣 (20) 台東縣, 臺東縣 (21) 花蓮縣 (22) 宜蘭縣 (23) 澎湖縣 (24) 金門縣 (25) 連江縣

A. THE 36 UNARY DATASETS

Japanese Dataset: disney-movies

(1) 白雪姫 (2) ピノキオ (3) ファンタジア (4) ダンボ (5) バンビ (6) ラテン・アメリカの旅 (7) 三人の騎士 (8) メイク・マイン・ミュージック (9) ファン・アンド・ファンシー・フリー (10) メロディ・タイム (11) イカボードとトード氏 (12) シンデレラ (13) ふしぎの国のアリス, 不思議の国のアリス (14) ピーター・パン (15) わんわん物語 (16) 眠れる森の美女 (17) 101 匹わんちゃん (18) 王様の剣 (19) ジャングル・ブック (20) おしゃれキャット (21) ロビン・フッド (22) くまのプーさん (23) ビアンカの大冒険 (24) きつねと猟犬 (25) コルドロン (26) オリビアちゃんの大冒険 (27) オリバー・ニューヨーク子猫物語 (28) リトル・マーメイド (29) ビアンカの大冒険ゴールデン・イーグルを救え (30) 美女と野獣 (31) アラジン (32) ライオン・キング (33) ポカホンタス (34) ノートルダムの鐘 (35) ヘラクレス (36) ムーラン (37) ターザン (38) ファンタジア 2000, ファンタジア２０００ (39) ラマになった王様 (40) アトランティス 失われた帝国 (41) リロ・アンド・スティッチ (42) トレジャー・プラネット (43) ブラザー・ベア (44) ホーム・オン・ザ・レンジ (45) チキン・リトル (46) ルイスと未来泥棒 (47) アメリカン・ドッグ (48) ラパンゼル・アンブライデド, ラプンツェル, ラパンゼル (49) ザ・フロッグ・プリンセス

Japanese Dataset: constellations

(1) アンドロメダ座, アンドロメダ (2) ポンプ座, ポンプ (3) ふうちょう座, ふうちょう (4) みずがめ座, みずがめ, 水瓶座, 水瓶 (5) わし座, わし (6) さいだん座, さいだん (7) おひつじ座, おひつじ, 牡羊座, 牡羊 (8) ぎょしゃ座, ぎょしゃ (9) うしかい座, うしかい (10) ちょうこくぐ座, ちょうこくぐ (11) きりん座, きりん (12) かに座, かに, 蟹座, 蟹 (13) りょうけん座, りょうけん (14) おおいぬ座, おおいぬ (15) こいぬ座, こいぬ (16) やぎ座, やぎ, 山羊座, 山羊 (17) りゅうこつ座, りゅうこつ (18) カシオペヤ座, カシオペア座, カシオペヤ, カシオペア (19) ケンタウルス座, ケンタウルス (20) ケフェウス座, ケフェウス (21) くじら座, くじら (22) カメレオン座, カメレオン (23) コンパス座, コンパス (24) はと座, はと (25) かみのけ座, かみのけ (26) みなみのかんむり座, みなみのかんむり (27) かんむり座, かんむり (28) からす座, からす (29) コップ座, コップ (30) みなみじゅうじ座, みなみじゅうじ (31) はくちょう座, はくちょう (32) いるか座, いるか (33) かじき座, かじき (34) りゅう座, りゅう (35) こうま座, こうま (36) エリダヌス座, エリダヌス (37) ろ座, ろ (38) ふたご座, ふたご, 双子座, 双子 (39) つる座, つる (40) ヘルクレス座, ヘルクレス (41) とけい座, とけい (42) うみへび座, うみへび (43) みずへび座, みずへび (44) インディアン座, インディアン (45) とかげ座, とかげ (46) しし座, しし, 獅子座, 獅子 (47) こじし座, こじし (48) うさぎ座, うさぎ (49) てんびん座, てんびん, 天秤座, 天秤 (50) おおかみ座, おおかみ (51) やまねこ座, やまねこ (52) こと座, こと (53) テーブルさん座, テーブルさん (54) けんびきょう座, けんびきょう (55) いっかくじゅう座, いっかくじゅう (56) はえ座, はえ (57) じょうぎ座, じょうぎ (58) はちぶんぎ座, はちぶんぎ (59) へびつかい座, へびつかい (60) オリオン座, オリオン (61) くじゃく座, くじゃく (62) ペガスス座, ペガスス (63) ペルセウス座, ペルセウス (64) ほうおう座, ほうおう (65) がか座, がか (66) うお座, うお, 魚座, 魚 (67) みなみのうお座, みなみのうお (68) とも座, とも (69) らしんばん座, らしんばん (70) レチクル座, レチクル (71) や座, や (72) いて座, いて, 射手座, 射手 (73) さそり座, さそり, 蠍座, 蠍 (74) ちょうこくしつ座, ちょうこくしつ (75) たて座, たて (76) へび座, へび (77) ろくぶんぎ座, ろくぶんぎ (78) おうし座, おうし, 牡牛座, 牡牛 (79) ぼうえんきょう座, ぼうえんきょう (80) さんかく座, さんかく (81) みなみのさんかく座, みなみのさんかく (82) きょしちょう座, きょしちょう (83) おおぐま座, おおぐま (84) こぐま座, こぐま (85) ほ座, ほ (86) おとめ座, おとめ, 乙女座, 乙女 (87) とびうお座, とびうお (88) こぎつね座, こぎつね

Japanese Dataset: countries

(1) アフガニスタン, アフガニスタン・イスラム共和国 (2) アルバニア, アルバニア共和国 (3) アルジェリア, アルジェリア民主人民共和国 (4) アンドラ, アンドラ公国 (5) アンゴラ, アンゴラ共和国 (6) アンティグア・バーブーダ (7) アルゼンチン, アルゼンチン共和国 (8) アルメニア, アルメニア共和国 (9) オーストラリア, オーストラリア連邦, 豪州, 豪, 濠 (10) オーストリア, オーストリア共和国, オーストリー, 墺 (11) アゼルバイジャン, アゼルバイジャン共和国 (12) バハマ, バハマ国 (13) バーレーン, バーレーン王国 (14) バングラデシュ, バングラデシュ人民共和国 (15) バルバドス (16) ベラルーシ, ベラルーシ共和国 (17) ベルギー, ベルギー王国 (18) ベリーズ (19) ベナン, ベナン共和国 (20) ブータン, ブータン王国 (21) ボリビア, ボリビア共和国 (22) ボスニア・ヘルツェゴビナ (23) ボツワナ, ボツワナ共和国 (24) ブラジル, ブラジル連邦共和国, 伯 (25) ブルネイ, ブルネイ・ダルサラーム国 (26) ブルガリア, ブルガリア共和国 (27) ブルキナファソ (28) ブルンジ, ブルンジ共和国 (29) カンボジア, カンボジア王国 (30) カメルーン, カメルーン共和国 (31) カナダ, 加 (32) カーボベルデ, カーボベルデ共和国 (33) 中央アフリカ, 中央アフリカ共和国 (34) チャド, チャド共和国 (35) チリ, チリ共和国 (36) 中国, 中華人民共和国, チャイナ (37) コロンビア, コロンビア共和国 (38) コモロ, コモロ連合 (39) コンゴ, コンゴ共和国 (40) コンゴ, コンゴ民主共和国 (41) コスタリカ, コスタリカ共和国 (42) コートジボワール, コートジボワール共和国 (43) クロアチア, クロアチア共和国 (44) キューバ, キューバ共和国 (45) キプロス, キプロス共和国 (46) チェコ, チェコ共和国 (47) デンマーク, デンマーク王国 (48) ジブチ, ジブチ共和国 (49) ドミニカ, ドミニカ連邦, ドミニカ国 (50) ドミニカ共和国 (51) 東ティモール, 東ティモール民主共和国 (52) エクアドル, エクアドル共和国 (53) エジプト, エジプト・アラブ共和国 (54) エルサルバドル, エルサルバドル共和国 (55) 赤道ギニア, 赤道ギニア共和国 (56) エリトリア, エリトリア国 (57) エストニア, エストニア共和国 (58) エチオピア, エチオピア連邦民主共和国 (59) フィジー, フィジー諸島共和国 (60) フィンランド, フィンランド共和国 (61) フランス, フランス共和国, 仏 (62) ガボン, ガボン共和国 (63) ガンビア, ガンビア共和国 (64) グルジア (65) ドイツ, ドイツ連邦共和国, 独 (66) ガーナ, ガーナ共和国 (67) ギリシャ, ギリシャ共和国, 希 (68) グレナダ (69) グアテマラ, グアテマラ共和国 (70) ギニア, ギニア共和国 (71) ギニアビサウ, ギニアビサウ共和国 (72) ガイアナ, ガイアナ協同共和国 (73) ハイチ, ハイチ共和国 (74) ホンジュラス, ホンジュラス共和国 (75) ハンガリー, ハンガリー共和国 (76) アイスランド, アイスランド共和国 (77) インド, 印 (78) インドネシア, インドネシア共和国 (79) イラン, イラン・イスラム共和国 (80) イラク, イラク共和国 (81) アイルランド (82) イスラエル, イスラエル国 (83) イタリア, イタリア共和国, 伊 (84) ジャマイカ (85) 日本, 日本国 (86) ヨルダン, ヨルダン・ハシミテ王国 (87) カザフスタン, カザフスタン共和国 (88) ケニア, ケニア共和国 (89) キリバス, キリバス共和国 (90) 北朝鮮, 朝鮮民主主義人民共和国, 朝鮮 (91) 韓国, 大韓民国 (92) クウェート, クウェート国 (93) キルギス, キルギス共和国, キルギスタン (94) ラオス, ラオス人民民主共和国 (95) ラトビア, ラトビア共和国 (96) レバノン, レバノン共和国 (97) レソト, レソト王国 (98) リベリア, リベリア共和国 (99) リビア, 大リビア・アラブ社会主義人民ジャマーヒリーヤ国 (100) リヒテンシュタイン, リヒテンシュタイン公国 (101) リトアニア, リトアニア共和国 (102) ルクセンブルク, ルクセンブルク大公国 (103) マケドニア, マケドニア・旧ユーゴスラビア共和国 (104) マダガスカル, マダガスカル共和国 (105) マラウイ, マラウィ, マラウイ共和国 (106) マレー, マレーシア, 馬来, 馬 (107) モルディブ, モルディブ共和国 (108) マリ, マリ共和国 (109) マルタ, マルタ共和国 (110) マーシャル諸島, マーシャル諸島共和国 (111) モーリタニア, モーリタニア・イスラム共和国 (112) モーリシャス, モーリシャス共和国 (113) メキシコ, メキシコ合衆国, 墨 (114) ミクロネシア, ミクロネシア連邦 (115) モルドバ,

モルドバ共和国 (116) モナコ, モナコ公国 (117) モンゴル, モンゴル国, 蒙古, 蒙 (118) モンテネグロ, モンテネグロ共和国 (119) モロッコ, モロッコ王国 (120) モザンビーク, モザンビーク共和国 (121) ミャンマー, ミャンマー連邦 (122) ナミビア, ナミビア共和国 (123) ナウル, ナウル共和国 (124) ネパール, ネパール王国 (125) オランダ, オランダ王国, 蘭 (126) ニュージーランド (127) ニカラグア, ニカラグア共和国 (128) ニジェール, ニジェール共和国 (129) ナイジェリア, ナイジェリア連邦共和国 (130) ノルウェー, ノルウェー王国 (131) オマーン, オマーン国 (132) パキスタン, パキスタン・イスラム共和国 (133) パラオ, パラオ共和国 (134) パレスチナ, パレスチナ国 (135) パナマ, パナマ共和国 (136) パプアニューギニア, パプアニューギニア独立国 (137) パラグアイ, パラグアイ共和国 (138) ペルー, ペルー共和国, 秘 (139) フィリピン, フィリピン共和国, 比 (140) ポーランド, ポーランド共和国 (141) ポルトガル, ポルトガル共和国, 葡 (142) カタール, カタール国 (143) ルーマニア (144) ロシア, ロシア連邦, 露 (145) ルワンダ, ルワンダ共和国 (146) セントキッツ・ネイビス, セントクリストファー・ネイビス (147) セントルシア (148) セントビンセント・グレナディーン, セントビンセントおよびグレナディーン諸島 (149) サモア, サモア独立国 (150) サンマリノ, サンマリノ共和国 (151) サントメ・プリンシペ, サントメ・プリンシペ民主共和国 (152) サウジアラビア, サウジアラビア王国 (153) セネガル, セネガル共和国 (154) セルビア, セルビア共和国 (155) セーシェル, セーシェル共和国 (156) シエラレオネ, シエラレオネ共和国 (157) シンガポール, シンガポール共和国 (158) スロバキア, スロバキア共和国 (159) スロベニア, スロベニア共和国 (160) ソロモン諸島 (161) ソマリア, ソマリア民主共和国 (162) 南アフリカ, 南アフリカ共和国, 南ア (163) エスパニア, スペイン, エスパーニャ, 西 (164) スリランカ, スリランカ民主社会主義共和国 (165) スーダン, スーダン共和国 (166) スリナム, スリナム共和国 (167) スワジランド, スワジランド王国 (168) スウェーデン, スウェーデン王国, 瑞典 (169) スイス, スイス連邦, 瑞西, 瑞 (170) シリア, シリア・アラブ共和国 (171) 台湾, 中華民国, 中華台北, チャイニーズタイペイ (172) タジキスタン, タジキスタン共和国 (173) タンザニア, タンザニア連合共和国 (174) タイ, タイ王国, 泰 (175) トーゴ, トーゴ共和国 (176) トンガ, トンガ王国 (177) トリニダード・トバゴ, トリニダード・トバゴ共和国 (178) チュニジア, チュニジア共和国 (179) トルコ, トルコ共和国, 土 (180) トルクメニスタン (181) ツバル (182) ウガンダ, ウガンダ共和国 (183) ウクライナ (184) アラブ首長国連邦 (185) イギリス, グレートブリテンおよび北アイルランド連合王国, 英国 (186) アメリカ, アメリカ合衆国, 米国, 米, 合衆国 (187) ウルグアイ, ウルグアイ東方共和国 (188) ウズベキスタン, ウズベキスタン共和国 (189) バヌアツ, バヌアツ共和国 (190) バチカン, バチカン市国 (191) ベネズエラ, ベネズエラ・ボリバル共和国 (192) ベトナム, ベトナム社会主義共和国, 越南, 越 (193) 西サハラ, サハラ・アラブ民主共和国 (194) イエメン, イエメン共和国 (195) ザンビア, ザンビア共和国 (196) ジンバブエ, ジンバブェ共和国

Japanese Dataset: mlb-teams

(1) アトランタ・ブレーブス, ブレーブス (2) フロリダ・マーリンズ, マーリンズ (3) ニューヨーク・メッツ, メッツ (4) フィラデルフィア・フィリーズ, フィリーズ (5) ワシントン・ナショナルズ, ナショナルズ (6) シカゴ・カブス, カブス (7) シンシナティ・レッズ, レッズ (8) ヒューストン・アストロズ, アストロズ (9) ミルウォーキー・ブリュワーズ, ミルウォーキー・ブルワーズ, ブリュワーズ, ブルワーズ (10) ピッツバーグ・パイレーツ, パイレーツ (11) セントルイス・カージナルス, カージナルス (12) アリゾナ・ダイヤモンドバックス, ダイヤモンドバックス (13) コロラド・ロッキーズ, ロッキーズ (14) ロサンゼルス・ドジャース, ブルックリン・ドジャース, ドジャース (15) サンディエゴ・パドレス, パドレス (16) サンフランシスコ・ジャイアンツ, ニューヨーク・ジャイアンツ, ジャイアンツ (17) ボルチモア・オリオールズ, ボルティモア・オリオールズ, オリオールズ (18) ボストン・レッドソックス, レッドソックス (19) ニューヨーク・ヤンキース, ヤンキース (20) タンパベイ・デビルレイズ, デビルレイズ, レイズ (21) トロント・ブルージェイズ, ブルージェイズ (22) シカゴ・ホワイトソックス, ホワイトソックス (23) クリーブランド・インディアンス, クリーブランド・インディアンズ, インディアンス, インディアンズ (24) デトロイト・タイガース, タイガース (25) カンザスシティ・ロイヤルズ, カンザスシティー・ロイヤルズ, ロイヤルズ (26) ミネソタ・ツインズ, ツインズ (27) ロサンゼルス・エンゼルス, ロサンジェルス・エンジェルス, ロサンゼルス・エンジェルス, アナハイム・エンジェルス, アナハイム・エンゼルス, エンゼルス, エンジェルス, アナハイム (28) オークランド・アスレチックス, アスレチックス (29) シアトル・マリナーズ, マリナーズ (30) テキサス・レンジャーズ, レンジャーズ

Japanese Dataset: nba-teams

(1) ボストン・セルティックス, セルティックス (2) ニュージャージー・ネッツ, ネッツ (3) ニューヨーク・ニックス, ニックス (4) フィラデルフィア・セブンティシクサーズ, セブンティシクサーズ (5) トロント・ラプターズ, ラプターズ (6) シカゴ・ブルズ, ブルズ (7) クリーブランド・キャバリアーズ, キャバリアーズ (8) デトロイト・ピストンズ, ピストンズ (9) インディアナ・ペイサーズ, インディアナ・ペーサーズ, ペイサーズ, ペーサーズ (10) ミルウォーキー・バックス, バックス (11) アトランタ・ホークス, ホークス (12) シャーロット・ボブキャッツ, ボブキャッツ (13) マイアミ・ヒート, ヒート (14) オーランド・マジック, マジック (15) ワシントン・ウィザーズ, ウィザーズ (16) ダラス・マーベリックス, マーベリックス (17) ヒューストン・ロケッツ, ロケッツ (18) メンフィス・グリズリーズ, バンクーバー・グリズリーズ, グリズリーズ (19) ニューオーリンズ・ホーネッツ, ニューオリンズ・ホーネッツ, ホーネッツ (20) サンアントニオ・スパーズ, スパーズ (21) デンバー・ナゲッツ, ナゲッツ (22) ミネソタ・ティンバーウルブズ, ティンバーウルブズ (23) ポートランド・トレイルブレイザーズ, ポートランド・ブレイザーズ, トレイルブレイザーズ, ブレイザーズ (24) オクラホマシティ・サンダー, サンダー, シアトル・スーパーソニックス, スーパーソニックス (25) ユタ・ジャズ, ジャズ (26) ゴールデンステート・ウォリアーズ, ゴールデンステイト・ウォリアーズ, ウォリアーズ (27) ロサンゼルス・クリッパーズ, クリッパーズ (28) ロサンゼルス・レイカーズ, ロサンゼルス・レイカース, レイカーズ, レイカース, レーカーズ (29) フェニックス・サンズ, サンズ (30) サクラメント・キングス, サクラメント・キングズ, キングス, キングズ

Japanese Dataset: nfl-teams

(1) バッファロー・ビルズ, ビルズ (2) マイアミ・ドルフィンズ, ドルフィンズ (3) ニューイングランド・ペイトリオッツ, ペイトリオッツ (4) ニューヨーク・ジェッツ, ジェッツ (5) ボルチモア・レイブンズ, ボルチモア・レイヴィンズ, ボルティモア・レイヴンズ, ボルチモア・レイヴンズ, ボルティモア・レイヴァンズ, レイブンズ, レイヴィンズ, レイヴンズ (6) シンシナティ・ベンガルズ, ベンガルズ (7) クリーブランド・ブラウンズ, ブラウンズ (8) ピッツバーグ・スティーラーズ, スティーラーズ (9) ヒューストン・テキサンズ, テキサンズ (10) インディアナポリス・コルツ, コルツ (11) ジャクソンビル・ジャガーズ, ジャガーズ (12) テネシー・タイタンズ, タイタンズ (13) デンバー・ブロンコズ, デンヴァー・ブロンコス, デンバー・ブロンコス, ブロンコズ, ブロンコス (14) カンザスシティ・チーフス, チーフス (15) オークランド・レイダーズ, オークランド・レイダース, レイダーズ, レイダース (16) サンディエゴ・チャージャーズ, サ

ンディエゴ・チャージャース, チャージャーズ, チャージャース (17) ダラス・カウボーイズ, カウボーイズ (18) ニューヨーク・ジャイアンツ, ジャイアンツ (19) フィラデルフィア・イーグルス, イーグルス (20) ワシントン・レッドスキンズ, レッドスキンズ (21) シカゴ・ベアーズ, ベアーズ (22) デトロイト・ライオンズ, ライオンズ (23) グリーンベイ・パッカーズ, パッカーズ (24) ミネソタ・バイキングス, ミネソタ・バイキングズ, ミネソタ・ヴァイキングス, バイキングス, ヴァイキングス (25) アトランタ・ファルコンズ, ファルコンズ (26) カロライナ・パンサーズ, パンサーズ (27) ニューオーリンズ・セインツ, ニューオリンズ・セインツ, セインツ (28) タンパベイ・バッカニアーズ, バッカニアーズ (29) アリゾナ・カージナルス, アリゾナ・カーディナルズ, カージナルス, カーディナルズ, カーディナルス (30) セントルイス・ラムズ, ラムズ (31) サンフランシスコ・フォーティナイナーズ, フォーティナイナーズ (32) シアトル・シーホークス, シーホークス

Japanese Dataset: car-makers

(1) アキュラ (2) アルファ・ロメオ, アルファロメオ (3) アストンマーチン, アストンマーティン, アストン・マーティン (4) アウディ, アウディー (5) ベントレー (6) BMW, ビー・エム・ダブリュー (7) ビュイック (8) キャディラック, キャデラック (9) シボレー, シェビー (10) クライスラー (11) シトロエン (12) 大宇 (13) ダイハツ (14) ダッジ (15) フェラーリ (16) フィアット (17) フォード (18) ジオ (19) 本田, ホンダ (20) ハマー (21) 現代, ヒュンダイ, 現代自動車 (22) インフィニティ (23) いすゞ, いすゞ自動車, イスズ, いすず (24) ジャガー (25) ジープ (26) 起亜, 起亜自動車 (27) ランボルギーニ (28) ランチア, ランチャ (29) ランドローバー (30) レクサス (31) リンカーン (32) ロータス, ロータス・カーズ (33) マセラティ (34) マツダ (35) メルセデス・ベンツ, メルセデスベンツ (36) マーキュリー (37) ミニ, MINI (38) 三菱, みつびし, 三菱自動車, ミツビシ (39) 日産, にっさん, 日産自動車, ニッサン (40) オールズモビル (41) オペル (42) プジョー (43) プリムス (44) ポンティアック, ポンテアック (45) ポルシェ (46) ルノー (47) ロールス・ロイス, ロールスロイス (48) サーブ (49) サターン (50) セアト (51) スマート (52) スバル (53) スズキ, 鈴木 (54) トヨタ, 豊田, トヨタ自動車 (55) フォルクスワーゲン (56) ボルボ

Japanese Dataset: us-presidents

(1) ジョージ・ワシントン, ワシントン (2) ジョン・アダムズ, ジョン・アダムス, アダムス (3) トーマス・ジェファーソン, トーマス・ジェファソン, トマス・ジェファーソン, トマス・ジェファソン, ジェファーソン, ジェファソン (4) ジェームズ・マディスン, ジェームズ・マディンソン, ジェームス・マディソン, ジェームズ・マディソン, マディスン, マディンソン, マディソン (5) ジェームズ・モンロー, ジェームス・モンロー, モンロー (6) ジョン・クィンシー・アダムズ, ジョン・クインシー・アダムス, クィンシー・アダムズ, アダムズ, アダムス (7) アンドリュー・ジャクソン, アンドルー・ジャクソン, ジャクソン (8) マーティン・ヴァンビューレン, マーティン・バン・ビューレン, マーチン・バン・ビューレン, マーティン・ヴァン・ビューレン, ヴァン・ビューレン, ビューレン (9) ウィリアム・ハリソン, ウイリアム・ハリソン, ウィリアム・ヘンリー・ハリソン, ウィリアム・Ｈ・ハリソン, ウィリアム・H・ハリソン, ハリソン (10) ジョン・タイラー, タイラー (11) ジェームズ・ポーク, ジェームス・ポーク, ジェームズ・Ｋ・ポーク, ポーク (12) ザカリー・テイラー, ザカリー・タイラー, ザカリー・テーラー, テイラー, タイラー, テーラー (13) ミラード・フィルモア, フィルモア (14) フランクリン・ピアース, フランクリン・ピアス, ピアース, ピアス (15) ジェームズ・ブキャナン, ジェームス・ブキャナン, ブキャナン (16) エイブラハム・リンカーン, エイブ, リンカーン (17) アンドルー・ジョンソン, アンドリュー・ジョンソン, ジョンソン (18) ユリシーズ・グラント, ユリシーズ・Ｓ・グラント, ユリシーズ・S・グラント, グラント (19) ラザフォード・ヘイズ, ラザフォード・Ｂ・ヘイズ, ラザフォード・B・ヘイズ, ヘイズ (20) ジェームズ・ガーフィールド, ジェームズ・Ａ・ガーフィールド, ジェームス・ガーフィールド, ガーフィールド (21) チェスター・アーサー, チェスター・Ａ・アーサー, チェスター・A・アーサー, アーサー (22) グロバー・クリーブランド, グローバー・クリーブランド, スティーブン・クリーブランド, クリーブランド (23) ベンジャミン・ハリソン, ハリソン (24) ウィリアム・マッキンリー, ウィリアム・マッキンレー, ウイリアム・マッキンリー, マッキンレー, マッキンリー (25) セオドア・ルーズベルト, セオドア・ローズヴェルト, ルーズベルト, ローズヴェルト (26) ウィリアム・タフト, ウイリアム・タフト, ウィリアム・Ｈ・タフト, ウィリアム・H・タフト, ウィリアム・ハワード・タフト, タフト (27) ウッドロウ・ウィルソン, ウッドロー・ウィルソン, トーマス・ウッドロー・ウィルソン, トマス・ウッドロー・ウィルソン, ウィルソン (28) ウォレン・ハーディング, ウォーレン・ハーディング, ウォーレン・Ｇ・ハーディング, ウオレン・G・ハーディング, ハーディング (29) カルビン・クーリッジ, カルヴァン・クーリッジ, クーリッジ (30) ハーバート・フーヴァー, ハーバート・フーバー, ハーバート・Ｃ・フーバー, ハーバート・C・フーヴァー, ハーバート・クラーク・フーヴァー, フーバー, フーヴァー (31) フランクリン・ルーズベルト, フランクリン・Ｄ・ルーズベルト, フランクリン・D・ルーズベルト, フランクリン・ローズヴェルト, ルーズベルト, ローズヴェルト (32) ハリー・トルーマン, ハリー・Ｓ・トルーマン, ハリー・S・トルーマン, トルーマン (33) ドワイト・アイゼンハワー, ドワイト・Ｄ・アイゼンハワー, ドワイト・D・アイゼンハワー, アイク, アイゼンハワー (34) ジョン・Ｆ・ケネディ, ジョン・ケネディ, ジョン・F・ケネディ, ジョン, ケネディ (35) リンドン・ジョンソン, リンドン・Ｂ・ジョンソン, リンドン, ジョンソン (36) リチャード・ニクソン, リチャード・Ｍ・ニクソン, リチャード・M・ニクソン, ディック, ニクソン (37) ジェラルド・フォード, ジェラルド・Ｒ・フォード, ジェラルド・R・フォード, ジェリー, フォード (38) ジミー・カーター, ジミー・カータ, ジェームズ・E・カーター, ジミー, カータ, カーター (39) ロナルド・レーガン, ロナルド・Ｗ・レーガンロナルド・W・レーガン, ロン, レーガン (40) ジョージ・ブッシュ, ジョージ・H・W・ブッシュ, ジョージ・Ｈ・Ｗ・ブッシュ, ブッシュ (41) ビル・クリントン, ウイリアム・クリントン, ウィリアム・Ｊ・クリントン, ウィリアム・ジェファーソン・クリントン, ウィリアム・J・クリントン, ビル, クリントン (42) ジョージ・Ｗ・ブッシュ, ジョージ・W・ブッシュ, ジョージ・ウォーカー・ブッシュ, ブッシュ

Japanese Dataset: us-states

(1) アラスカ州, アラスカ (2) アラバマ州, アラバマ (3) アーカンソー州, アーカンソー (4) アリゾナ州, アリゾナ (5) カリフォルニア州, カリフォルニア (6) コロラド州, コロラド (7) コネチカット州, コネチカット (8) デラウェア州, デラウェア (9) フロリダ州, フロリダ (10) ジョージア州, ジョージア (11) ハワイ州, ハワイ (12) アイオワ州, アイオワ (13) アイダホ州, アイダホ (14) イリノイ州, イリノイ (15) インディアナ州, インディアナ (16) カンザス州, カンザス (17) ケンタッキー州, ケンタッキー (18) ルイジアナ州, ルイジアナ (19) マサチューセッツ州, マサチューセッツ (20) メリーランド州, メリーランド (21) メイン州, メイン (22) ミシガン州, ミシガン (23) ミネソタ州, ミネソタ (24) ミズーリ州, ミズーリ (25) ミシシッピ州, ミシシッピ (26) モンタナ州, モンタナ (27) ノースカロライナ州, ノースカロライナ (28) ノースダコタ州, ノースダコタ (29) ネブラスカ州, ネブラスカ (30) ニューハンプシャー州, ニューハンプシャー (31) ニュージャージ

ー州, ニュージャージー (32) ニューメキシコ州, ニューメキシコ (33) ネバダ州, ネバダ (34) ニューヨーク州, ニューヨーク (35) オハイオ州, オハイオ (36) オクラホマ州, オクラホマ (37) オレゴン州, オレゴン (38) ペンシルバニア州, ペンシルバニア (39) ロードアイランド州, ロードアイランド (40) サウスカロライナ州, サウスカロライナ (41) サウスダコタ州, サウスダコタ (42) テネシー州, テネシー (43) テキサス州, テキサス (44) ユタ州, ユタ (45) バージニア州, バージニア (46) バーモント州, バーモント (47) ワシントン州, ワシントン (48) ウィスコンシン州, ウィスコンシン (49) ウェストバージニア州, ウェストバージニア (50) ワイオミング州, ワイオミング

Japanese Dataset: japan-emperors

(1) 神武, 神武天皇 (2) 綏靖, 綏靖天皇 (3) 安寧, 安寧天皇 (4) 懿徳, 懿徳天皇 (5) 孝昭, 孝昭天皇 (6) 孝安, 孝安天皇 (7) 孝霊, 孝霊天皇 (8) 孝元, 孝元天皇 (9) 開化, 開化天皇 (10) 崇神, 崇神天皇 (11) 垂仁, 垂仁天皇 (12) 景行, 景行天皇 (13) 成務, 成務天皇 (14) 仲哀, 仲哀天皇 (15) 応神, 応神天皇 (16) 仁徳, 仁徳天皇 (17) 履中, 履中天皇 (18) 反正, 反正天皇 (19) 允恭, 允恭天皇 (20) 安康, 安康天皇 (21) 雄略, 雄略天皇 (22) 清寧, 清寧天皇 (23) 顕宗, 顕宗天皇 (24) 仁賢, 仁賢天皇 (25) 武烈, 武烈天皇 (26) 継体, 継体天皇 (27) 安閑, 安閑天皇 (28) 宣化, 宣化天皇 (29 欽明, 欽明天皇 (30) 敏達, 敏達天皇 (31) 用明, 用明天皇 (32) 崇峻, 崇峻天皇 (33) 推古, 推古天皇 (34) 舒明, 舒明天皇 (35) 皇極, 皇極天皇 (36 孝徳, 孝徳天皇 (37) 斉明, 斉明天皇 (38) 天智, 天智天皇 (39) 弘文, 弘文天皇 (40) 天武, 天武天皇 (41) 持統, 持統天皇 (42) 文武, 文武天皇 (43 元明, 元明天皇 (44) 元正, 元正天皇 (45) 聖武, 聖武天皇 (46) 孝謙, 孝謙天皇 (47) 淳仁, 淳仁天皇 (48) 称徳, 称徳天皇 (49) 光仁, 光仁天皇 (50 桓武, 桓武天皇 (51) 平城, 平城天皇 (52) 嵯峨, 嵯峨天皇 (53) 淳和, 淳和天皇 (54) 仁明, 仁明天皇 (55) 文徳, 文徳天皇 (56) 清和, 清和天皇 (57 陽成, 陽成天皇 (58) 光孝, 光孝天皇 (59) 宇多, 宇多天皇 (60) 醍醐, 醍醐天皇 (61) 朱雀, 朱雀天皇 (62) 村上, 村上天皇 (63) 冷泉, 冷泉天皇 (64 円融, 円融天皇 (65) 花山, 花山天皇 (66) 一条, 一条天皇 (67) 三条, 三条天皇 (68) 後一条, 後一条天皇 (69) 後朱雀, 後朱雀天皇 (70) 後冷泉, 後冷泉天皇 (71) 後三条, 後三条天皇 (72) 白河, 白河天皇 (73) 堀河, 堀河天皇 (74) 鳥羽, 鳥羽天皇 (75) 崇徳, 崇徳天皇 (76) 近衛, 近衛天皇 (77) 後白河, 後白河天皇 (78) 二条, 二条天皇 (79) 六条, 六条天皇 (80) 高倉, 高倉天皇 (81) 安徳, 安徳天皇 (82) 後鳥羽, 後鳥羽天皇 (83) 土御門, 土御門天皇 (84) 順徳, 順徳天皇 (85) 仲恭, 仲恭天皇 (86) 後堀河, 後堀河天皇 (87) 四条, 四条天皇 (88) 後嵯峨, 後嵯峨天皇 (89) 後深草, 後深草天皇 (90) 亀山, 亀山天皇 (91) 後宇多, 後宇多天皇 (92) 伏見, 伏見天皇 (93) 後伏見, 後伏見天皇 (94) 後二条, 後二条天皇 (95) 花園, 花園天皇 (96) 後醍醐, 後醍醐天皇 (97) 後村上, 後村上天皇 (98) 長慶, 長慶天皇 (99) 後亀山, 後亀山天皇 (100) 後小松, 後小松天皇 (101) 光厳, 光厳天皇 (102) 光明, 光明天皇 (103) 崇光, 崇光天皇 (104) 後光厳, 後光厳天皇 (105) 後円融, 後円融天皇 (106) 後小松, 後小松天皇 (107) 称光, 称光天皇 (108) 後花園, 後花園天皇 (109) 後土御門, 後土御門天皇 (110) 後柏原, 後柏原天皇 (111) 後奈良, 後奈良天皇 (112) 正親町, 正親町天皇 (113) 後陽成, 後陽成天皇 (114) 後水尾, 後水尾天皇 (115) 明正, 明正天皇 (116) 後光明, 後光明天皇 (117) 後西, 後西天皇 (118) 霊元, 霊元天皇 (119) 東山, 東山天皇 (120) 中御門, 中御門天皇 (121) 桜町, 桜町天皇 (122) 桃園, 桃園天皇 (123) 後桜町, 後桜町天皇 (124) 後桃園, 後桃園天皇 (125) 光格, 光格天皇 (126) 仁孝, 仁孝天皇 (127) 孝明, 孝明天皇 (128) 明治, 明治天皇 (129) 大正, 大正天皇 (130) 昭和, 昭和天皇 (131) 今上, 今上天皇

Japanese Dataset: japan-prime-ministers

(1) 伊藤博文 (2) 黒田清隆, 黒田清隆 (3) 山縣有朋, 山県有朋 (4) 松方正義 (5) 大隈重信 (6) 桂太郎 (7) 西園寺公望 (8) 山本權兵衞, 山本権兵衛 (9) 寺内正毅 (10) 原敬 (11) 高橋是清 (12) 加藤友三郎 (13) 清浦奎吾 (14) 加藤高明 (15) 若槻禮次郎, 若槻礼次郎 (16) 田中義一 (17) 濱口雄幸 (18) 犬養毅 (19) 齋藤實 (20) 岡田啓介 (21) 廣田弘毅, 広田弘毅 (22) 林銑十郎 (23) 近衞文麿, 近衛文麿 (24) 平沼騏一郎 (25) 阿部信行 (26) 米内光政 (27) 東條英機 (28) 小磯國昭 (29) 東久邇宮稔彦王 (30) 鈴木貫太郎 (31) 幣原喜重郎 (32) 吉田茂 (33) 片山哲 (34) 芦田均 (35) 鳩山一郎 (36) 石橋湛山 (37) 岸信介 (38) 池田勇人 (39) 佐藤榮作, 佐藤栄作 (40) 田中角榮, 田中角栄 (41) 三木武夫 (42) 福田赳夫 (43) 大平正芳 (44) 鈴木善幸 (45) 中曾根康弘, 中曽根康弘 (46) 竹下登 (47) 宇野宗佑 (48) 海部俊樹 (49) 宮澤喜一 (50) 細川護熙, 細川護煕 (51) 羽田孜 (52) 村山富市 (53) 橋本龍太郎 (54) 小渕恵三 (55) 森喜朗 (56) 小泉純一郎 (57) 安倍晋三 (58) 福田康夫 (59) 麻生太郎

Japanese Dataset: japan-provinces

(1) 北海道 (2) 青森県, 青森 (3) 岩手県, 岩手 (4) 宮城県, 宮城 (5) 秋田県, 秋田 (6) 山形県, 山形 (7) 福島県, 福島 (8) 茨城県, 茨城 (9) 栃木県, 栃木 (10) 群馬県, 群馬 (11) 埼玉県, 埼玉 (12) 千葉県, 千葉 (13) 東京都, 東京 (14) 神奈川県, 神奈川 (15) 富山県, 富山 (16) 石川県, 石川 (17) 福井県, 福井 (18) 山梨県, 山梨 (19) 長野県, 長野 (20) 新潟県, 新潟 (21) 岐阜県, 岐阜 (22) 静岡県, 静岡 (23) 愛知県, 愛知 (24) 三重県, 三重 (25) 滋賀県, 滋賀 (26) 京都府, 京都 (27) 大阪府, 大阪 (28) 兵庫県, 兵庫 (29) 奈良県, 奈良 (30) 和歌山県, 和歌山 (31) 鳥取県, 鳥取 (32) 島根県, 島根 (33) 岡山県, 岡山 (34) 広島県, 広島 (35) 山口県, 山口 (36) 徳島県, 徳島 (37) 香川県, 香川 (38) 愛媛県, 愛媛 (39) 高知県, 高知 (40) 福岡県, 福岡 (41) 佐賀県, 佐賀 (42) 長崎県, 長崎 (43) 熊本県, 熊本 (44) 大分県, 大分 (45) 宮崎県, 宮崎 (46) 鹿児島県, 鹿児島 (47) 沖縄県, 沖縄

Appendix B

The 5 Binary Datasets

English Dataset: us-governors

(1) Alabama/Bob Riley (2) Alaska/Sarah Palin, Alaska/Frank Murkowski (3) American Samoa/Togiola Tulafono, American Samoa/Togiola T.A. Tulafono (4) Arizona/Janet Napolitano, Arizona/Jan Brewer (5) Arkansas/Mike Beebe, Arkansas/Mike Huckabee (6) California/Arnold Schwarzenegger (7) Colorado/Bill Ritter, Colorado/Bill Owens, Colorado/Bill Ritter Jr (8) Connecticut/M. Jodi Rell, Connecticut/Jodi Rell, Connecticut/Mary Jodi Rell (9) Delaware/Jack Markell, Delaware/Ruth Ann Minner, Delaware/Ruth Minner (10) District of Columbia/Adrian Fenty (11) Florida/Charlie Crist, Florida/Jeb Bush (12) Georgia/Sonny Perdue (13) Guam/Felix Camacho, Guam/Felix Perez Camacho (14) Hawaii/Linda Lingle (15) Idaho/Butch Otter, Idaho/C.L. Butch Otter, Idaho/C.L. Otter, Idaho/Dirk Kempthorne (16) Illinois/Pat Quinn, Illinois/Rod Blagojevich, Illinois/Rod R. Blagojevich (17) Indiana/Mitch Daniels (18) Iowa/Chet Culver, Iowa/Tom Vilsack (19) Kansas/Kathleen Sebelius (20) Kentucky/Steve Beshear, Kentucky/Steven Beshear, Kentucky/Steven L. Beshear, Kentucky/Ernie Fletcher (21) Louisiana/Bobby Jindal, Louisiana/Kathleen Blanco, Louisiana/Kathleen Babineaux Blanco (22) Maine/John Baldacci, Maine/John E. Baldacci (23) Maryland/Martin O'malley, Maryland/Robert Ehrlich (24) Massachusetts/Deval Patrick, Massachusetts/Mitt Romney (25) Michigan/Jennifer M. Granholm, Michigan/Jennifer Granholm (26) Minnesota/Tim Pawlenty (27) Mississippi/Haley Barbour (28) Missouri/Jay Nixon, Missouri/Jeremiah Nixon, Missouri/Matt Blunt (29) Montana/Brian Schweitzer (30) Nebraska/Dave Heineman, Nebraska/David Heineman (31) Nevada/Jim Gibbons, Nevada/James Gibbons, Nevada/Kenny Guinn (32) New Hampshire/John Lynch (33) New Jersey/Jon Corzine, New Jersey/John Corzine (34) New Mexico/Bill Richardson (35) New York/David Paterson, New York/David A. Paterson, New York/Eliot Spitzer, New York/George Pataki, New York/George E. Pataki (36) North Carolina/Beverly Perdue, North Carolina/Bev Perdue, North Carolina/Mike Easley, North Carolina/Michael Easley, North Carolina/Michael F. Easley (37) North Dakota/John Hoeven (38) Northern Mariana Islands/Benigno Fitial, Northern Mariana Islands/Benigno R. Fitial (39) Ohio/Ted Strickland, Ohio/Bob Taft (40) Oklahoma/Brad Henry, Oklahoma/Brad Jones (41) Oregon/Ted Kulongoski (42) Pennsylvania/Ed Rendell, Pennsylvania/Edward Rendell, Pennsylvania/Edward G. Rendell (43) Puerto Rico/Luis Fortuno, Puerto Rico/Anibal acevedo vila (44) Rhode Island/Donald Carcieri, Rhode Island/Donald L. Carcieri, Rhode Island/Don Carcieri (45) South Carolina/Mark Sanford (46) South Dakota/Mike Rounds, South Dakota/M. Michael Rounds (47) Tennessee/Phil Bredesen (48) Texas/Rick Perry (49) United States Virgin Islands/John De Jongh, US Virgin Islands/John de Jongh, Virgin Islands/John deJongh Jr, U.S. Virgin Islands/John de Jongh (50) Utah/Jon Huntsman, Utah/Jon Huntsman Jr (51) Vermont/Jim Douglas, Vermont/James H. Douglas, Vermont/James Douglas (52) Virginia/Tim Kaine (53) Washington/Christine Gregoire, Washington/Chris Gregoire (54) West Virginia/Joe Manchin III, West Virginia/Joe Manchin (55) Wisconsin/Jim Doyle, Wisconsin/James Doyle (56) Wyoming/Dave Freudenthal, Wyoming/David Freudenthal

B. THE 5 BINARY DATASETS

Chinese Dataset: taiwan-mayors

(1) 桃園縣/朱立倫 (2) 新竹市/林政則 (3) 台中市/胡志強 (4) 彰化縣/卓伯源, 彰化縣/翁金珠 (5) 花蓮縣/謝深山, 花蓮縣/張福興 (6) 南投縣/李朝卿, 南投縣/林宗男 (7) 宜蘭縣/呂國華, 宜蘭縣/劉守成, 宜蘭縣/陳定南 (8) 台中縣/黃仲生 (9) 苗栗縣/劉政鴻, 苗栗縣/傅學鵬 (10) 澎湖縣/王乾發, 澎湖縣/賴峰偉 (11) 台北縣/周錫瑋, 台北縣/蘇貞昌 (12) 新竹縣/鄭永金 (13) 高雄縣/楊秋興 (14) 台東縣/鄺麗貞, 台東縣/徐慶元, 台東縣/吳俊立 (15) 台南縣/蘇煥智 (16) 嘉義縣/陳明文 (17) 基隆市/張通榮, 基隆市/許財利 (18) 台南市/許添財 (19) 嘉義市/黃敏惠, 嘉義市/陳麗貞 (20) 屏東縣/曹啓鴻, 屏東縣/蘇嘉全 (21) 雲林縣/蘇治芬, 雲林縣/張榮味 (22) 金門縣/李炷烽 (23) 連江縣/陳雪生 (24) 台北市/郝龍斌, 台北市/馬英九 (25) 高雄市/陳菊

Chinese-English Dataset: nba-teams

(1) 波士頓塞爾提克/Boston Celtics, 波士頓賽爾提克/Boston Celtics, 波士頓塞爾蒂克/Boston Celtics, 波士頓凱爾特人/Boston Celtics, 塞爾提克/Boston Celtics, 賽爾提克/Boston Celtics, 塞爾蒂克/Boston Celtics, 凱爾特人/Boston Celtics (2) 紐澤西籃網/New Jersey Nets, 新澤西籃網/New Jersey Nets, 籃網/New Jersey Nets (3) 紐約尼克/New York Knicks, 紐約尼客/New York Knicks, 紐約尼克斯/New York Knicks, 尼克/New York Knicks, 尼客/New York Knicks, 尼克斯/New York Knicks (4) 費城 76 人/Philadelphia 76ers, 費城 76 人/Philadelphia 76ers, 費城七十六人/Philadelphia 76ers, 費城七六人/Philadelphia 76ers, 76 人/Philadelphia 76ers, 七十六人/Philadelphia 76ers, 七六人/Philadelphia 76ers, 費城七六人/Philadelphia Sixers (5) 多倫多暴龍/Toronto Raptors, 暴龍/Toronto Raptors, 猛龍/Toronto Raptors (6) 芝加哥公牛/Chicago Bulls, 公牛/Chicago Bulls (7) 克里夫蘭騎士/Cleveland Cavaliers, 克裡夫蘭騎士/Cleveland Cavaliers, 克利夫蘭騎士/Cleveland Cavaliers, 騎士/Cleveland Cavaliers (8) 底特律活塞/Detroit Pistons, 活塞/Detroit Pistons (9) 印第安那溜馬/Indiana Pacers, 印地安那溜馬/Indiana Pacers, 印地安納溜馬/Indiana Pacers, 印第安納溜馬/Indiana Pacers, 溜馬/Indiana Pacers, 步行者/Indiana Pacers (10) 密爾瓦基公鹿/Milwaukee Bucks, 公鹿/Milwaukee Bucks, 雄鹿/Milwaukee Bucks (11) 亞特蘭大老鷹/Atlanta Hawks, 老鷹/Atlanta Hawks (12) 夏洛特山貓/Charlotte Bobcats, 山貓/Charlotte Bobcats (13) 邁阿密熱火/Miami Heat, 熱火/Miami Heat (14) 奧蘭多魔術/Orlando Magic, 魔術/Orlando Magic (15) 華盛頓巫師/Washington Wizards, 華盛頓奇才/Washington Wizards, 巫師/Washington Wizards, 奇才/Washington Wizards (16) 達拉斯小牛/Dallas Mavericks, 小牛/Dallas Mavericks (17) 休士頓火箭/Houston Rockets, 休斯頓火箭/Houston Rockets, 火箭/Houston Rockets (18) 孟斐斯灰熊/Memphis Grizzlies, 孟菲斯灰熊/Memphis Grizzlies, 曼菲斯灰熊/Memphis Grizzlies, 曼斐斯灰熊/Memphis Grizzlies, 溫哥華灰熊/Memphis Grizzlies, 灰熊/Memphis Grizzlies (19) 紐奧良黃蜂/New Orleans Hornets, 紐奧良黃蜂/New Orleans/Oklahoma City Hornets, 夏洛特黃蜂/Charlotte Hornets, 黃蜂/New Orleans Hornets, 黃蜂/New Orleans/Oklahoma City Hornets, 黃蜂/Charlotte Hornets (20) 聖安東尼奧馬刺/San Antonio Spurs, 聖安東尼市馬刺/San Antonio Spurs, 聖安東尼馬刺/San Antonio Spurs, 馬刺/San Antonio Spurs (21) 丹佛金塊/Denver Nuggets, 丹佛掘金/Denver Nuggets, 金塊/Denver Nuggets, 掘金/Denver Nuggets (22) 明尼蘇達灰狼/Minnesota Timberwolves, 明尼蘇達木狼/Minnesota Timberwolves, 明尼蘇達森林狼/Minnesota Timberwolves, 灰狼/Minnesota Timberwolves, 森林狼/Minnesota Timberwolves (23) 波特蘭拓荒者/Portland Trail Blazers, 拓荒者/Portland Trail Blazers, 開拓者/Portland Trail Blazers, 波特蘭拓荒者/Portland Blazers (24) 奧克拉荷馬雷霆/Oklahoma City Thunder, 雷霆/Oklahoma City Thunder, 西雅圖超音速/Seattle Supersonics, 超音速/Seattle Supersonics (25) 猶他爵士/Utah Jazz, 爵士/Utah Jazz (26) 金州勇士/Golden State Warriors, 勇士/Golden State Warriors (27) 洛杉磯快艇/Los Angeles Clippers, 快艇/Los Angeles Clippers, 快船/Los Angeles Clippers (28) 洛杉磯湖人/Los Angeles Lakers, 湖人/Los Angeles Lakers (29) 鳳凰城太陽/Phoenix Suns, 太陽/Phoenix Suns (30) 沙加緬度國王/Sacramento Kings, 國王/Sacramento Kings

English Dataset: fed-agencies

(1) ABMC/American Battle Monuments Commission (2) ACF/Administration for Children and Families (3) ACHP/Advisory Council on Historic Preservation (4) ADD/Administration on Developmental Disabilities (5) ADF/African Development Foundation (6) AFIS/American Forces Information Service (7) AFRICOM/United States African Command (8) AHRQ/Agency for Healthcare Research and Quality (9) AID/Agency for International Development, USAID/Agency for International Development (10) AMS/Agricultural Marketing Service (11) AMTRAK/National Railroad Passenger Corporation, NRPC/National Railroad Passenger Corporation (12) ANA/Administration for Native Americans (13) AOA/Administration on Aging (14) AOC/Architect of the Capitol (15) APHIS/Animal and Plant Health Inspection Service (16) ARB/Administrative Review Board (17) ARC/Appalachian Regional Commission (18) ARS/Agricultural Research Service (19) ATF/Bureau of Alcohol (20) ATSDR/Agency for Toxic Substances and Disease Registry, TSDR/Agency for Toxic Substances and Disease Registry (21) BBG/Broadcasting Board of Governors (22) BCBP/Bureau of Customs and Border Protection (23) BCIS/Bureau of Citizenship and Immigration Services (24) BEA/Bureau of Economic Analysis (25) BEP/Bureau of Engraving and Printing (26) BIA/Bureau of Indian Affairs (27) BIS/Bureau of Industry and Security (28) BJA/Bureau of Justice Assistance (29) BJS/Bureau of Justice Statistics (30) BLM/Bureau of Land Management (31) BLS/Bureau of Labor Statistics (32) BOP/Bureau of Prisons, BOP/Federal Bureau of Prisons (33) BPA/Bonneville Power Administration (34) BPAI/Board of Patent Appeals and Interferences (35) BPD/Bureau of Public Debt, BPD/Bureau of the Public Debt (36) BRB/Benefits Review Board (37) BTS/Bureau of Transportation Statistics (38) CB/Children's Bureau (39) CBO/Congressional Budget Office (40) CBP/Customs and Border Protection (41) CCB/Child Care Bureau (42) CCR/Commission on Civil Rights, USCCR/Commission on Civil Rights (43) CDBG/Community Development Block Grants (44) CDC/Centers for Disease Control and Prevention, CDC/Centers for Disease Control (45) CDFI/Community Development Financial Institution Fund (46) CEA/Council of Economic Advisers (47) CEN/Bureau of the Census (48) CIA/Central Intelligence Agency (49) CENTCOM/Central Command (50) CEQ/Council on Environmental Quality (51) CFA/Commission of Fine Arts (52) CFBCI/Center for Faith-Based and Community Initiatives (53) CFOC/Chief Financial Officers Council (54) CFTC/Commodity Futures Trading Commission (55) CMS/Centers for Medicare and Medicaid Services (56) CNCS/Corporation for National and Community Service, CNS/Corporation for National Service (57) CNCS/Corporation for National and Community Service (58) CNPP/Center for Nutrition Policy and Promotion (59) COPS/Community Oriented Policing Services (60) CPD/Community Planning and Development (61) CPSC/Consumer Product Safety Commission (62)

CREES/Cooperative State Research (63) CRS/Congressional Research Service (64) CSB/Chemical Safety and Hazard Investigation Board, USCSB/Chemical Safety and Hazard Investigations Board (65) CSCE/Commission on Security and Cooperation in Europe (66) CSOSA/Court Services and Offender Supervision Agency (67) DA/Department of the Army, USA/Department of the Army, USA/United States Army (68) DARPA/Defense Advanced Research Projects Agency (69) DAU/Defense Acquisition University (70) DCAA/Defense Contract Audit Agency (71) DCMA/Defense Contract Management Agency (72) DEA/Drug Enforcement Administration, DEA/Drug Enforcement Agency (73) DeCA/Defense Commissary Agency (74) DFAS/Defense Finance and Accounting Service (75) DHS/Department of Homeland Security, DHS/Homeland Security Department (76) DIA/Defense Intelligence Agency (77) DISA/Defense Information Systems Agency (78) DLA/Defense Logistics Agency (79) DLSA/Defense Legal Services Agency (80) DNFSB/Defense Nuclear Facilities Safety Board (81) DNI/Director of National Intelligence, ODNI/Director of National Intelligence (82) DOC/Department of Commerce (83) DOD/Department of Defense, DOD/Defense Department, DefenseLINK/Department of Defense (84) DODEA/Department of Defense Education Activity (85) DOE/Department of Energy, DOE/Energy Department, Energy/Department of Energy (86) DOI/Department of the Interior (87) DOJ/Department of Justice, USDOJ/United States Department of Justice, USDOJ/Department of Justice (88) DOL/Department of Labor, DOL/Labor Department, Labor/Department of Labor (89) DOS/Department of State, State/Department of State (90) DOT/Department of Transportation, DOT/Transportation Department (91) DPC/Domestic Policy Council (92) DRBC/Delaware River Basin Commission (93) DSCA/Defense Security Cooperation Agency (94) DSS/Defense Security Service (95) DTIC/Defense Technical Information Center (96) DTRA/Defense Threat Reduction Agency (97) EBSA/Employee Benefits Security Administration (98) ECA/Bureau of Educational and Cultural Affairs (99) ECAB/Employees' Compensation Appeals Board (100) ED/Department of Education, ED/Education Department, Education/Department of Education, DOE/Department of Education (101) EDA/Economic Development Administration (102) EE/Energy Efficiency and Renewable Energy (103) EEOC/Equal Employment Opportunity Commission (104) EIA/Energy Information Administration (105) EOIR/Executive Office for Immigration Review (106) EOUSA/Executive Office for U.S. Attorneys (107) EPA/Environmental Protection Agency, EPA/United States Environmental Protection Agency (108) ERIC/Education Resources Information Center, ERIC/Educational Resources Information Center (109) ERS/Economic Research Service (110) ESA/Economics and Statistics Administration, ESA/Employment Standards Administration (111) ETA/Employment and Training Administration (112) EUCOM/European Command (113) ExIm/Export-Import Bank of the United States, Ex-Im Bank/Export-Import Bank of the United States, EXIMBANK/Export-Import Bank of the United States (114) FAA/Federal Aviation Administration (115) FAS/Foreign Agricultural Service (116) FASAB/Federal Accounting Standards Advisory Board (117) FBI/Federal Bureau of Investigation (118) FBOP/Federal Bureau of Prisons (119) FCA/Farm Credit Administration (120) FCC/Federal Communications Commission (121) FCIC/Federal Citizen Information Center (122) FCIC/Federal Crop Insurance Corporation (123) FCSC/Foreign Claims Settlement Commission of the United States (124) FDA/Food and Drug Administration (125) FDIC/Federal Deposit Insurance Corporation (126) FEB/Federal Executive Boards, FEB's/Federal Executive Boards (127) FEC/Federal Election Commission (128) FEMA/Federal Emergency Management Agency (129) FERC/Federal Energy Regulatory Commission (130) FFB/Federal Financing Bank (131) FGDC/Federal Geographic Data Committee (132) FHA/Federal Housing Administration (133) FHEO/Office of Fair Housing and Equal Opportunity (134) FHFB/Federal Housing Finance Board (135) FHWA/Federal Highway Administration (136) FICE/Federal Interagency Committee on Education (137) FinCEN/Financial Crimes Enforcement Network (138) FJC/Federal Judicial Center (139) FLETC/Federal Law Enforcement Training Center (140) FLRA/Federal Labor Relations Authority (141) FMC/Federal Maritime Commission (142) FMCS/Federal Mediation and Conciliation Service (143) FMCSA/Federal Motor Carrier Safety Administration (144) FMS/Financial Management Service (145) FMSHRC/Federal Mine Safety and Health Review Commission, FMSHRC/Federal Mine Safety (146) FNS/Food and Nutrition Service (147) FRA/Federal Railroad Administration (148) FRB/Federal Reserve Board (149) FRS/Federal Reserve System, FED/Federal Reserve System, The Fed/Federal Reserve System (150) FRTIB/Federal Retirement Thrift Investment Board (151) FS/Forest Service (152) FSA/Farm Service Agency (153) FSA/Office of Federal Student Aid (154) FSIS/Food Safety and Inspection Service (155) FTA/Federal Transit Administration (156) FTC/Federal Trade Commission (157) FWS/Fish and Wildlife Service (158) FYSB/Family and Youth Services Bureau (159) GAO/Government Accountability Office, GAO/General Accounting Office (160) GIPSA/Grain Inspection, Packers and Stockyards Administration, GIPSA/Grain Inspection (161) GNMA/Government National Mortgage Association, Ginnie Mae/Government National Mortgage Association (162) GPO/Government Printing Office (163) GSA/General Services Administration (164) HCFA/Health Care Financing Administration (165) HHS/Department of Health and Human Services, HHS/Health and Human Services Department, DHHS/Department of Health and Human Services (166) HMI/Healthy Marriage Initiative (167) HRSA/Health Resources and Services Administration (168) HSB/Head Start Bureau (169) HUD/Department of Housing and Urban Development, HUD/Housing and Urban Development Department, HUD/Housing and Urban Development, HUD/Housing Office (170) IACB/Indian Arts and Crafts Board (171) IAF/Inter-American Foundation (172) IBB/International Broadcasting Bureau (173) ICAF/Industrial College of the Armed Forces (174) ICC/Interstate Commerce Commission (175) ICE-FPS/United States Federal Protective Service (176) ICE/Immigration and Customs Enforcement, ICE/U.S. Immigration and Customs Enforcement (177) IES/Institute of Education Sciences (178) IHS/Indian Health Service (179) ILAB/Bureau of International Labor Affairs (180) IMLS/Institute of Museum and Library Services (181) INS/Immigration and Naturalization Service (182) IO/Bureau of International Organization Affairs (183) IRS/Internal Revenue Service (184) ITA/International Trade Administration (185) ITC/International Trade Commission, USITC/United States International Trade Commission, USITC/International Trade Commission (186) JCS/Joint Chiefs of Staff, JCSLink/Joint Chiefs of Staff (187) JFCOM/Joint Forces Command (188) JFSC/Joint Forces Staff College (189) LC/Library of Congress, LOC/Library of Congress (190) LIHEAP/Low Income Home Energy Assistance Program (191) LOC/Library of Congress (192) LSA/Learn and Serve America (193) LSC/Legal Services Corporation (194) MA/Maritime Administration, MARAD/Maritime Administration (195) MARAD/Maritime Administration (196) MBDA/Minority Business Development Agency (197) MCC/Millennium Challenge Corporation (198) MDA/Missile Defense Agency (199) MedPAC/Medicare Payment Advisory Commission (200) MMC/Marine Mammal Commission (201) MMS/Minerals Management Service (202) MSHA/Mine Safety and Health Administration (203) MSPB/Merit Systems Protection Board (204) NAEP/National Assessment of Educational Progress (205) NAL/National Agricultural Library (206) NARA/National Archives and Records Administration (207) NASA/National Aeronautics and Space Administration (208) NASS/National Agricultural Statistics Service (209) NCA/National Cemetery Administration (210) NCD/National Council on Disability (211) NCES/National Center for Education Statistics (212) NCIS/Naval Criminal Investigative Service (213) NCJRS/National Criminal Justice Reference Service (214) NCPC/National Capital Planning Commission (215) NCS/National Communications System (216) NCUA/National

Credit Union Administration (217) NDIC/National Drug Intelligence Center (218) NDU/National Defense University (219) NEA/National Endowment for the Arts (220) NEC/National Economic Council (221) NEH/National Endowment for the Humanities (222) NESDIS/National Environmental Satellite (223) NGA/National Geospatial-Intelligence Agency, NGIA/National Geospatial-Intelligence Agency (224) NGS/National Geodetic Survey (225) NHTSA/National Highway Traffic Safety Administration (226) NIC/National Ice Center (227) NIC/National Institute of Corrections (228) NIDRR/National Institute on Disability and Rehabilitation Research (229) NIFC/National Interagency Fire Center (230) NIFL/National Institute for Literacy (231) NIGC/National Indian Gaming Commission (232) NIH/National Institutes of Health (233) NIJ/National Institute of Justice (234) NIMA/National Imagery and Mapping Agency (235) NIMH/National Institute of Mental Health (236) NIST/National Institute of Standards and Technology (237) NLM/National Library of Medicine (238) NLRB/National Labor Relations Board (239) NMB/National Mediation Board (240) NMMR/National Mine Map Repository (241) NNSA/National Nuclear Security Administration (242) NOAA/National Oceanic and Atmospheric Administration (243) NORTHCOM/Northern Command (244) NOS/National Ocean Service, NOS/National Oceanic Service (245) NPS/National Park Service, ParkNet/National Park Service (246) NPTO/National Petroleum Technology Office (247) NRC/Nuclear Regulatory Commission (248) NRCS/Natural Resources Conservation Service (249) NRO/National Reconnaissance Office (250) NSA/National Security Agency (251) NSC/National Security Council (252) NSF/National Science Foundation (253) NTIA/National Telecommunications and Information Administration (254) NTID/National Technical Institute for the Deaf (255) NTIS/National Technical Information Service (256) NTRC/National Transportation Research Center (257) NTSB/National Transportation Safety Board (258) NWC/National War College (259) NWS/National Weather Service (260) NWTRB/Nuclear Waste Technical Review Board (261) OA/Office of Administration (262) OCC/Office of the Comptroller of the Currency (263) OCO/Office of Communications and Outreach (264) OCR/Office for Civil Rights (265) OCS/Office of Community Services Block Grant (266) OCSE/Office of Child Support Enforcement, CSE/Office of Child Support Enforcement (267) ODS/Office of the Deputy Secretary (268) OELA/Office of English Language Acquisition (269) OESE/Office of Elementary and Secondary Education (270) OFA/Office of Family Assistance (271) OFCCP/Office of Federal Contract Compliance Programs (272) OFHEO/Office of Federal Housing Enterprise Oversight (273) OGC/Office of the General Counsel (274) OGE/Office of Government Ethics, USOGE/United States Office of Government Ethics, USOGE/U.S. Office of Government Ethics (275) OIA/Office of Insular Affairs (276) OIG/Office of the Inspector General, USPS-OIG/Office of the Inspector General, OIG/Office of Inspector General (277) OII/Office of Innovation and Improvement (278) OJP/Office of Justice Programs (279) OLCA/Office of Legislation and Congressional Affairs (280) OLMS/The Office of Labor-Management Standards (281) OMB/Office of Management and Budget (282) OMH/Office of Minority Health (283) ONCHIT/Office of the National Coordinator for Health Information Technology (284) ONDCP/Office of National Drug Control Policy (285) OPE/Office of Postsecondary Education (286) OPIC/Overseas Private Investment Corporation (287) OPM/Office of Personnel Management (288) ORNL/Oak Ridge National Laboratory (289) ORR/Office of Refugee Resettlement (290) OS/Office of the Secretary (291) OSC/Office of Special Counsel (292) OSDFS/Office of Safe and Drug Free Schools (293) OSEP/Office of Special Education Programs (294) OSERS/Office of Special Education and Rehabilitative Services (295) OSHA/Occupational Safety and Health Administration, OSHA/Occupational Safety (296) OSHRC/Occupational Safety and Health Review Commission (297) OSM/Office of Surface Mining (298) OSTI/Office of Scientific and Technical Information (299) OSTP/Office of Science and Technology Policy (300) OTP/Office of Technology Policy (301) OTS/Office of Thrift Supervision (302) OUS/Office of the Under Secretary (303) OVAE/Office of Vocational and Adult Education (304) OVC/Office for Victims of Crime (305) OWCP/Office of Workers' Compensation Programs (306) PACOM/Pacific Command (307) PBGC/Pension Benefit Guaranty Corporation (308) PC/Peace Corps (309) PCPID/President's Committee for People with Intellectual Disabilities (310) PFPA/Pentagon Force Protection Agency (311) PHMSA/Pipeline and Hazardous Materials Safety Administration (312) PIH/Public and Indian Housing, PIH/Office of Public and Indian Housing (313) PRC/Postal Regulatory Commission (314) PSC/Program Support Center (315) PTO/Patent and Trademark Office, USPTO/Patent and Trademark Office (316) PWBA/Pension and Welfare Benefits Administration (317) RBS/Rural Business-Cooperative Service (318) RD/Rural Development (319) RFA/Radio Free Asia (320) RFE/Radio Free Europe (321) RHS/Rural Housing Service (322) RITA/Research and Innovative Technology Administration (323) RMA/Risk Management Agency (324) RRB/Railroad Retirement Board (325) RSA/Rehabilitation Services Administration (326) RUS/Rural Utilities Service (327) SAMHSA/Substance Abuse and Mental Health Services Administration (328) SBA/Small Business Administration (329) SEC/Securities and Exchange Commission (330) SEC/Securities Exchange Commission (331) SI/Smithsonian Institution (332) SJI/State Justice Institute (333) SLAC/Stanford Linear Accelerator Center (334) SLSDC/Saint Lawrence Seaway Development Corporation (335) SNL/Sandia National Laboratories (336) SOCOM/Special Operations Command (337) SOUTHCOM/Southern Command, USSOUTHCOM/United States Southern Command (338) SRBC/Susquehanna River Basin Commission (339) SS/Secret Service (340) SSA/Social Security Administration (341) SSS/Selective Service System (342) STB/Surface Transportation Board (343) STRATCOM/Strategic Command, USSTRATCOM/Strategic Command (344) TA/Technology Administration (345) TANF/Temporary Assistance for Needy Families (346) TDA/Trade and Development Agency, USTDA/U.S. Trade and Development Agency, USTDA/United States Trade and Development Agency (347) TFI/Terrorism and Financial Intelligence (348) TIGTA/Treasury Inspector General for Tax Administration (349) TRANSCOM/Transportation Command (350) TSA/Transportation Security Administration (351) TTAB/Trademark Trial and Appeal Board (352) TTB/Alcohol and Tobacco Tax and Trade Bureau (353) TVA/Tennessee Valley Authority (354) USAF/Department of the Air Force (355) USAID/Agency for International Development, USAID/U.S. Agency for International Development (356) USARC/U.S. Arctic Research Commission (357) USBR/Bureau of Reclamation, BOR/Bureau of Reclamation (358) USCCR/US Commission on Civil Rights (359) USCG/United States Coast Guard, USCG/Coast Guard (360) USCIS/United States Citizenship and Immigration Services, USCIS/U.S. Citizenship and Immigration Services (361) USDA/Department of Agriculture, USDA/Agriculture Department, USDA/United States Department of Agriculture (362) USGS/United States Geological Survey, USGS/Geological Survey (363) USIA/United States Information Agency (364) USITC/U.S. International Trade Commission (365) USM/U.S. Mint (366) USMS/United States Marshals Service (367) USN/Department of the Navy, USN/Navy (368) USPIS/United States Postal Inspection Service (369) USPPD/United States Pentagon Police (370) USSAH/Armed Forces Retirement Home (371) USSC/United States Sentencing Commission, USSC/U.S. Sentencing Commission (372) USSS/United States Secret Service (373) USTR/United States Trade Representative (374) TREAS/Department of the Treasury, USTREAS/United States Department of the Treasury (375) USPS/United States Postal Service, USPS/Postal Service, USPS/U.S. Postal Service (376) VA/Department of Veterans Affairs, DVA/Department of Veterans Affairs, VA/Veterans Affairs Department (377) VBA/Veterans Benefits Administration (378) VETS/Veterans' Employment and Training Service (379) VHA/Veterans Health Administration (380) VOA/Voice of America (381) WAPA/Western Area Power Administration (382) WB/Women's Bureau (383) WHD/Wage and Hour Division, WH/Wage and Hour Division (384) WHIHBCU/President's Advisory Board on Historically Black Colleges and Universities (385) WHITCU/President's Advisory Board on Tribal Colleges and Universities (386) WHMO/White House Military Office (387) WHS/Washington Headquarters Services

English Dataset: car-makers

(1) Abarth/Italy, Abarth/Italia (2) Acura/Japan (3) Agrale Bus/Brasil (4) Alfa Romeo/Italy, Alfa Romeo/Italia (5) Alpina/Germany (6) Alvis/UK, Alvis/U.K, Alvis/United Kingdom (7) Apperson/USA, Apperson/United States (8) Aston Martin/UK, Aston Martin/U.K, Aston Martin/England, Aston Martin/United Kingdom (9) Audi/Germany (10) Austin Healey/UK, Austin Healey/U.K, Austin Healey/United Kingdom (11) Autobianchi/Italy, Autobianchi/Italia (12) Ballot/France (13) Bentley/UK, Bentley/U.K, Bentley/United Kingdom (14) Bertone/Italy, Bertone/Italia (15) Bmw/Germany (16) Bugatti/France (17) Buick/USA, Buick/United States (18) Cadillac/USA, Cadillac/United States (19) Callaway/USA, Callaway/United States (20) Caterham/UK, Caterham/U.K, Caterham/United Kingdom (21) Chevrolet/USA, Chevrolet/United States (22) Chrysler/USA, Chrysler/United States (23) Citroen/France (24) Dacia/Romania (25) Daewoo/Korea, Daewoo/South Korea (26) Daihatsu/Japan (27) DaimlerChrysler/Germany (28) Datsun/Japan (29) De Tomaso/Italy, De Tomaso/Italia, DeTomaso/Italy, DeTomaso/Italia (30) Dodge/USA, Dodge/United States (31) Eagle/USA, Eagle/United States (32) Ferrari/Italy, Ferrari/Italia, Ferrari Dino/Italy (33) Fiat/Italy, Fiat/Italia (34) Ford/USA, Ford/United States (35) Frazer Nash/UK, Frazer Nash/U.K, Frazer Nash/United Kingdom (36) General Motors/USA (37) Gillet/Belgium (38) Ginetta/UK, Ginetta/U.K, Ginetta/United Kingdom (39) Holden/Australia, GM Holden/Australia (40) GMC/USA, GMC/United States (41) Gurgel/Brasil (42) Honda/Japan (43) Hummer/USA, Hummer/United States (44) Hyundai/Korea, Hyundai/South Korea (45) Infiniti/Japan (46) Innocenti/Italy, Innocenti/Italia (47) Isdera/Germany (48) Isuzu/Japan (49) Italdesign/Italy, Italdesign/Italia (50) Iveco/Italy, Iveco/Italia (51) Jaguar/UK, Jaguar/U.K, Jaguar/United Kingdom, Jaguar/England (52) Jeep/USA, Jeep/United States (53) Kia/Korea, Kia/South Korea (54) Kissel/USA, Kissel/United States (55) Koenigsegg/Sweden (56) Lada/Russia (57) Lamborghini/Italy, Lamborghini/Italia (58) Lancia/Italy, Lancia/Italia (59) Land Rover/UK, Land Rover/U.K, Land Rover/England, Land Rover/United Kingdom, Rover/UK, Rover Group/U.K (60) Laraki/Morocco (61) Lexus/Japan (62) Ligier/France (63) Lincoln/USA, Lincoln/United States (64) Lotus/UK, Lotus/U.K, Lotus Cars/England, Lotus/United Kingdom, Lotus/England (65) Magna/Canada (66) Mahindra/India (67) Malaguti/Italy, Malaguti/Italia (68) Maserati/Italy, Maserati/Italia (69) Maybach/Germany (70) Mazda/Japan (71) McLaughlin/Canada (72) Mercedes/Germany, Mercedes Benz/Germany (73) Mercer/USA, Mercer/United States (74) Mercury/USA, Mercury/United States (75) MG/UK, MG/U.K, MG/United Kingdom, M.G/UK, MG/England (76) MINI/UK, MINI/U.K, Mini/United Kingdom (77) Mitsubishi/Japan (78) Morgan/UK, Morgan/U.K, Morgan/United Kingdom, Morgan Morot/UK (79) Nissan/Japan (80) Oldsmobile/USA, Oldsmobile/United States (81) Opel/Germany (82) Pagani/Italy, Pagani/Italia (83) Panoz/USA, Panoz/United States (84) Pegaso/Spain (85) Peugeot/France (86) Plymouth/USA, Plymouth/United States (87) Pontiac/USA, Pontiac/United States (88) Porsche/Germany, Porche/Germany (89) Proton/Malaysia (90) Reliant/UK, Reliant/U.K, Reliant/United Kingdom (91) Renault/France (92) Rolls Royce/UK, Rolls Royce/U.K, Rolls Royce/United Kingdom (93) RUF/Germany (94) Saab/Sweden (95) Saleen/USA, Saleen/United States (96) Saturn/USA, Saturn/United States (97) Scion/USA, Scion/United States (98) Seat/Spain (99) Shelby/USA, Shelby/United States (100) Skoda/Czech Republic, Skoda/Rep.Czech (101) Smart/Germany (102) Spyker/Netherlands (103) Ssangyong/Korea (104) Subaru Isuzu Automotive/USA (105) Subaru/Japan (106) Sunbeam/UK, Sunbeam/U.K, Sunbeam/United Kingdom (107) Suzuki/Japan (108) Talbot/France (109) Tata/India, Tata Motors/India (110) Toyota/Japan (111) Trabant/Germany (112) Triumph/UK, Triumph/U.K, Triumph/United Kingdom (113) Troller/Brasil (114) TVR/UK, TVR/U.K (115) Vauxhall/UK, Vauxhall/U.K, Vauxhall/United Kingdom (116) Vector/USA, Vector/United States (117) Venturi/France (118) Volkswagen/Germany, VW/Germany (119) Volvo/Sweden (120) Wartburg/Germany (121) Wiesmann/Germany (122) Yugo/Yugoslavia, Yugo/Jugoslavija

Appendix C

List Questions for TREC 13-15

List Questions for TREC 13

1.3 Which cities have Crip gangs?
2.3 What are titles of the group's releases?
3.3 In what countries was the Hale Bopp comet visible?
4.4 What movies did James Dean appear in?
5.5 What companies has AARP endorsed?
6.3 Famous people who have been Rhodes scholars.
6.4 What countries have Rhodes Scholars come from?
7.3 In what countries are agouti's found?
8.4 Who have been members of the organization? (Black Panthers)
9.1 Who are the members of Insane Clown Poose?
9.2 What albums has Insane Clown Poose made?
10.3 What diseases are prions associated with?
10.4 What researchers have worked with prions?
11.2 Who are Nirvana's band members?
11.5 What are Nirvana's albums?
15.1 Who are the members of the Rat Pack?
16.3 List doctors who have performed cateract surgery.
18.6 List the names of boxers Floyd Patterson fought.
20.2 What airlines have Concordes in their fleets?
21.2 List Club Med spots in the US
22.4 What books did he author? (Franz Kafka)
24.4 What prizes or awards has Frank Gehry won?
24.5 What buildings has Gehry designed?
26.5 What are names of Ice-T's albums?
30.5 What songs did Al Jolson Sing?
31.7 What movies did Jean Harlow appear in?
31.8 What leading men did Jean Harlow star opposite?
32.4 What festivals does Wicca have?
34.5 Name cities that have an Amtrak terminal.
36.4 Who were leaders of the Khmer Rouge?
37.2 List the Wiggles' Names.
37.4 List the Wiggles' Songs.
38.4 What are the different types of quarks?
39.3 List songs of The Clash
41.4 Who were the major players involved in the Teapot Dome scandal?
43.2 Categories of Nobel Prizes.
45.3 What countries has IFC financed projects in?
47.5 What schools did Bashar Assad attend?
48.4 In what countries as Abu Nidal operated from?
50.4 What planets will the Cassini space probe pass?
51.3 What other countries do Kurds live in?
52.5 What countries is Burger King located in?
53.4 What magazines does Conde Nast publish?
54.6 What schools did Eileen Marie Collins attend?
55.4 What books has Walter Mosley written?

56.3 List groups affected by Good Friday Agreement.
56.4 List key players in negotiating the G.F. Agreement.
58.2 List organizations A. Villar donated money to.
58.4 List companies in which Alberto Villar invested.
61.4 What countries does the Muslim Brotherhood operate in?
61.5 Name members of the Muslin Brotherhood group.
62.4 List members of the Berkman Center for Internet and Society
63.3 List states that have had problems with Boll Weevils
64.5 Where did Johnny Appliseed plant trees?
65.1 What are the names of the space shuttles?

List Questions for TREC 14

66.5 Which countries expressed regret about the loss [sinking of Kursk submarine]?
66.7 Which U.S. submarines were reportedly in the area [of Kursk sinking]?
67.6 Name other contestants [in Miss Universe 2000].
68.6 What were the names of the victims [of Port Arthur Massacre]?
68.7 What were the nationalities of the victims [of Port Arthur Massacre]?
69.7 Name players on the French team.
70.7 Who were on-ground witnesses to the accident [plane clipping cable wires in Italian resort]?
71.6 What countries besides U.S. fly F16s?
72.6 Who are some of the Bollywood stars?
73.6 In what countries could Viagra be obtained on the black market?
74.6 Name graduates of the [DePauw] university.
75.5 Name companies that are business competitors [of Merck].
75.7 Name products manufactured by Merck.
76.7 What movies was he [Bing Crosby] in?
77.6 Name opponents who Foreman defeated.
77.7 Name opponents who defeated Foreman.
78.7 What were some of his [Akira Kurosawa] Japanese film titles?
79.3 List students who were shot by Kip Kinkel
80.6 Identify nationalities of passengers on Flight 990
81.2 List other horses that have won the Kentucky Derby and Preakness but not the Belmont
82.3 Name the various puppets used in the "Howdy Doody Show"
82.4 Name the characters in the [Howdy Doody] show
83.4 Name the works of art that have been stolen from the Louvre
84.7 Provide a list of names/identifications given to meteorites
85.1 Name the ships of the NCL
85.6 Name the so-called theme cruises promoted by NCL
86.5 Name the children of Sani Abacha
87.4 List things named in honor of Enrico Fermi
88.4 In what foreign countries does the UPS operate?
89.3 What Little League teams have won the World Series?
90.1 What grape varieties are Virginia wines made from?
90.5 Name the Virginia Wine Festivals
91.3 Give the titles of Cliffs Notes Condensed Classics
92.3 What players has Arnold Palmer competed against in "The Skins Game"?
92.4 Which golf courses were designed by Arnold Palmer?
93.7 Who helped the candidates prepare [for the Bush Gore Debate]?
94.4 Who testified in defense of Susan McDougal?
95.5 What other countries formally congratulated China on the return of Hong Kong?
96.3 Who won gold medals in Nagano?
97.5 Crow's Record Titles
97.6 List the Crow's Band Members
98.5 List Legionaires
99.1 List Woody Guthrie's songs
100.7 Pitchers off of which Sosa homered
101.7 List Michael Weiss's competitors
102.6 List individuals associated with the Bag Dig
103.6 List players who scored touchdowns in the game [Super Bowl XXXIV]
104.4 List auto manufacturers in the show [1999 Auto Show-Detroit]
105.6 List names of eyewitnesses of eruption [1980 Mt. St. Helens]
106.6 Name the players in the [1998 World Series] Series
107.6 List dates of Chunnel closures
108.4 Name movies released by SPE (Sony Pictures Entertainment)
108.5 Name TV Shows by the SPE [Sony Pictures Entertainment]
109.5 Name companies involved in mergers with Telefonica of Spain
110.5 Name officials of the Club [Lions Club International]
110.6 Programs sponsored by the Lions Club [International]
111.4 Officals of the [Amway] Company

112.5 Corporation's [McDonald's] Top Officials
112.6 Non-hanburger restaurant holdings of the [McDonald's] Corporation
113.4 Name the camps started under his [Paul Newman] Hole in the Wall Foundation.
113.5 Name some of his [Paul Newman] movies
114.3 Various Occupations [of Jesse Ventura]
114.4 Movies/TV Shows [Jesse Ventura] appeared in
115.7 List personel of the [Longwood] Gardens
116.6 Who are some world leaders that have met there [Camp David]?
117.4 What other name is it [Kudzu] known by?
118.4 What medal [U.S. Medal of Honor] of honor recipients are in Congress?
119.4 What other products do they [Harley-Davidson] produce?
120.5 What awards has she [Rose Crumb] received?
121.3 What books did she [Rachel Carson] write?
122.7 What were some of his [Paul Revere] occupations?
123.5 What countries did Vicente Fox visit?
124.6 Who were Rocky Marciano's opponents?
125.1 In what operas has Caruso sung?
126.3 What official positions did Pope Pius XII hold before becomingPope?
127.6 People who attended the US Naval Academy in Annapolis, MD
128.3 What countries constitute the OPEC committee?
128.5 List of OPEC countries
129.4 Which countries were the original signers of NATO?
130.5 What countries have been struck by Tsunamis?
131.7 Individual who witnessed the Hindenburg Disaster
132.4 What posts has Kim Jong Il held in the government of this country?
133.3 As of the time of Hurricane Mitch, what previous hurricanes had higher death totals?
133.4 What countries offered aid for this hurricane [Mitch]?
134.2 List species whose genomes have been sequenced.
134.3 List the organizations that sequenced the Human genome.
135.5 What countries participated in this ["Food-for-Oil"] agreement by providing food or medicine?
136.7 What Shiite leaders were killed in Pakistan?
137.3 What other island groups are controlled by this government?
138.3 Where were UPU congresses held?
139.2 Which countries are members of the OIC?
139.3 Who has served as Secretary General of the OIC?
140.4 Employees of what companies are receiving benefits from this [PBGC] organization?

List Questions for TREC 15

141.6 Who have coached Moon in professional football?
141.7 List the professional teams for which Moon has been a player.
142.6 Name past and present LPGA commissioners.
142.7 Name tournaments in which LPGA players have participated.
143.6 Who have been "scholars" at the Institute?
143.7 Who have been "fellows" at the Institute?
144.6 In what conflicts has the division participated?
144.7 Who have commanded the division?
145.6 What defense and prosecution attorneys participated in the trial?
146.6 Which countries formally disapproved of the overthrow?
147.7 What individuals were at the wedding?
148.6 Tourists from which countries were among the dead?
149.7 What celebrities have appeared on The Daily Show?
150.7 Name supporting actors who performed in Cheers.
151.2 What races are part of the Winston Cup series?
151.3 Which drivers have won the Winston Cup?
152.4 List Mozart's operas.
153.4 List his movie nominations for best director.
154.6 List titles of movies, other than "Superman" movies, that Christopher Reeve acted in.
155.5 List countries visited by Chavez.
156.4 List winners of the NASCAR races.
157.6 Who has served as Secretary-General of the U.N.?
158.6 Name the schools of Tufts University.
159.3 List companies that have filed suits against Wal-Mart.
160.2 List the countries that have been provided loans by the IMF.
161.3 List the official sponsors of the game.
162.6 List facilities involved in the treatment of multiple myeloma.
163.5 List the artists represented in the collection.
164.2 What movies did she play in?
165.3 The Queen Mother received congratulatory greetings from what Heads of State?
166.6 What vaccines are known to be effective against avian flu?

C. LIST QUESTIONS FOR TREC 13-15

167.6 List the names of other millennium structures in England.

168.2 What charities have benefited from the sale or auction of his paintings?

169.5 What are the locations or names of other stone circles in the UK?

170.5 What artists has John Prine done duets with?

170.6 What are the titles of songs written by John Prine?

171.6 Name famous artists whose works have been purchased by Stephen Wynn or are displayed in his galleries.

172.7 Name unusual flavors created by Ben & Jerry's.

173.6 The WTO has held meetings in what countries?

173.7 Who has served as secretary general of the WTO?

174.6 What employees of the AFBF have been mentioned in the news?

175.6 What are the names of Elian's relatives?

176.1 Name cast members of the movie "An Officer and a Gentleman".

177.7 In what cities were the matches between Deep Blue and Kasparov held?

178.6 In what cities or towns have illegal methamphetamine labs been found?

179.5 Name movies Hedy Lamarr appeared in.

179.7 Name Hedy Lamarr's husbands.

180.5 Name members of the Lebanese Parliament.

181.6 Who were leading players for Manchester United in the 1990's?

181.8 Which British teams has Manchester United played?

182.3 What plays were performed at the 1998 Edinburgh Fringe?

183.4 What national leaders and spokespersons sent congratulatory messages following Thabo Mbeki's election as president of South Africa?

184.7 Name nations represented in the 1999 Chicago Marathon.

185.5 Name people who have won the Iditarod.

185.8 Which companies have sponsored the Iditarod?

186.3 What are the names of the three Great Pyramids?

186.7 Name additional pyramids of Egypt.

187.4 Name tributaries of the Amazon River.

188.3 What are the main commercial varieties of avocados?

188.4 What countries produce avocados?

189.1 What are the names of this author's books?

190.6 What food companies have been acquired by Heinz?

191.4 In what cities were International Rowing Federation Rowing World Cup events held?

192.3 What are some other Basque separatist groups?

193.7 What countries have donated to the WFP?

194.4 Who did Kasparov defeat in this tournament?

194.5 The purpose of this tournament was to help unify what world chess organizations?

195.6 What countries contributed troops to INTERFET?

196.3 Which European Union countries originally chose not to adopt the Euro?

197.4 What other mammals have been cloned from adult cells?

197.5 What countries have placed restrictions on human cloning research?

198.3 What other countries have signed contracts to work on this facility?

199.7 List other saints who have had the stigmata.

200.6 Name his children.

201.7 What plays did Shakespeare write?

202.4 What musicals did Cole Porter compose?

203.3 In what countries does Nissan manufacture vehicles outside of Japan?

204.5 Name elected government officials who are Mormon.

205.6 What other volcanoes are in the Philippines?

206.7 What other U.S. states have had dam failures?

207.7 Name other leaning towers.

208.5 In what cities has the Great Wall of China been found?

209.7 Who were her family members?

210.7 List universities that she visited.

211.7 What songs did she record?

212.7 List the songs he recorded.

213.4 What movies did Meg Ryan star in?

214.7 Who were the five finalists in the pageant?

215.7 List films shown at the 1999 Sundance Film Festival.