Data Analysis Project: Semi-Supervised Discovery of Named Entities and Relations from the Web

Sophie Wang

Advisor: Tom Mitchell

Machine Learning Department
Carnegie Mellon University

May 2009

Abstract:
This project studies semi-supervised discovery of named entities, relational entities and prepositional phrase attachments within a read-the-web framework. Meanings of an entity can be improvised and updated faster in the internet world than printed references. The main idea of this project is to study the feasibility of characterizing entities by web content directly. The approach is that contextual words around an entity on web pages are first extracted and converted into a Bag-Of-Word (BOW) representation. We then apply several supervised and semi-supervised learning methods on top of these contextual words for several well known research problems: Named Entities Recognition, Relation Extraction and Prepositional Phrase Attachment.

1 Introduction

The web has provided us a universal medium for data, information and knowledge exchange. We are proposing a breakthrough for artificial intelligence -- Read-the-Web (Mitchell, 2005) project, which is to automatically read billions web pages and harvest them into a knowledge repository. The main goal of the stage I for this project is to read natural language text and extract its factual contents. The optimistic feasibility comes from current internet, which contains a huge volume of general knowledge and great depth of ground facts, this rich and redundant factual content is our tremendous data to explore. Again, recent research progress has been made on natural language processing (NLP) by using machine learning algorithms. These state-of-art algorithms contribute metrics of statistical estimation on large data corpora. We have seen significant research works, such as Named Entity Recognition, Relation Extraction, and Word Sense Disambiguation etc by using supervised, semi-supervised or unsupervised learning methods. The confluence of these trends -- the tremendous web data, the progresses on natural language processes and machine learning, provides a promising perspective for our goal.

In this research, we use current web search engine (Google, Yahoo etc) to access web data. As we know, relevant information is very accessible by simply typing a query into a search engine. We can search for a person name, a school name, a product, a flight or hotel information etc. Queries are in any variety of forms, they can be a named entity, a phrase, or even a sentence, which are actually various items in natural language processing. In this sense, we are able to obtain relevant information for any NLP items through a search engine.

Consequently, we consider here the utility of a search-based Bag-Of-Word (BOW) representation for NLP items as a basis for a variety of NLP tasks, including named entity recognition, relation extraction, prepositional phrase etc. Our idea is inspired by the BOW model and the current web search engines. We characterize the probability distribution for any NLP item directly using the web. Given a named entity "MySpace" as an example, our main focus is to obtain a BOW vector, which represents a distribution of words surrounding "MySpace" on the web. To construct these BOW distributions for any text item, we use Google search engine to retrieve web pages. More precisely, we use an API provided by Google to automatically retrieve web-pages given "MySpace". We evaluate this BOW representation and apply it in three NLP research problems: Named Entity Recognition (NER), Relation Extraction (RE) and Preposition Phrase Attachment (PPA).

In the text below, we will give details how to create the Bag-of-Word (BOW) representation and go through several statistical analyses on BOW. In section 3, we briefly set forth the related work and problem definition for Named Entity Recognition, and also demonstrate the experiment results. We describe the work on Relation Extraction in section 4 and Prepositional Phrase Attachment in section 5. The last section summarizes the proposed methods and gives a conclusion.

2 BOW representation for NLP items

As we described above, our idea is inspired by the Bag-Of-Word model, which widely used in document classification. In this case, a document is represented as an unordered collection of words, disregarding grammar and word order. We extend this BOW model to represent a noun phrase and even any kind of NLP items, such as a named entity "Boston", a relational tuple "<Bill Gates, Microsoft>", a pattern "mayor of" and a prepositional phrase "with a telescope". Given the example "MySpace", we used surrounding texts that contain "MySpace" on the web documents to construct a BOW vector. In terms of feature space, a BOW of a named entity is a numeric feature vector defining the frequency of the word tokens co-occur with the given query in the surrounding text. Notice that the semantic meaning of a named entity dynamically changed over time. For instance, "Chicago" is a city name, became a famous band name in 1970s, and can be a movie name after 2002. As search engine company update their index files timely, our search-based BOW vector can be updated accordingly via the web search engine.

2.1 Constructing search-based BOW representation for any query

A BOW feature can be feasibly generated by counting co-occurrences of each word token in a given set of texts. Given a previous example "MySpace", we typed it as a query into Google Search Engine and retrieved a number of top ranked web pages, and then removed html markup from each web page and segmented it into sentences (Google Search API, WIT, OpenNLP). The sentences that contain the query "MySpace" were used to construct a BOW vector, which interpret a probability distribution of "MySpace" on the web. This is one of the three methods we used to generate a BOW vector (see details in section 2.2-1).

Figure 1 illustrates how a BOW vector looks like. The x-axis represents the word tokens appeared in the surrounding text of the given query "MySpace", and the y-axis is the counting frequency of tokens that co-occur with the query.

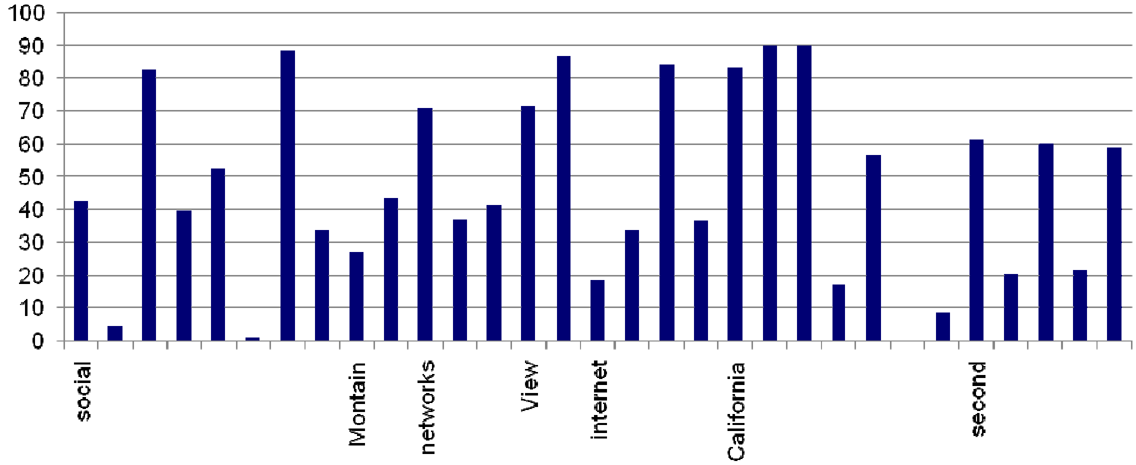


Figure 1. The illustration of a BOW vector of *"MySpace"*

2.2 Statistical analysis for the BOW feature

Several questions may emerge for our approach to constructing a BOW vector. For instance, what kind of surrounding texts to use, is it appropriate to use entire texts in a web page that contains "MySpace" to generate a BOW vector? There are thousands of web pages contain "MySpace", how many web pages to retrieve is optimal to generate a BOW vector? What is a comparison with other kind of vector features, such as spelling rules and contexts which are widely used in Named Entity Recognition? We will address these questions and statistically analyze them as follows.

1. What is an ideal surrounding contextual text to create a BOW vector?

This question is to analyze how many surrounding words to construct a BOW vectors. We have examined three methods to extract surrounding text in those retrieved web pages.

Method 1: Entire web page; use the entire page text (removing html makeup) that contains query to create a BOW vector.

Method 2: *k* character-length window (snippet); use query's left and right *k* surrounding characters to create a BOW vector. This is alike snippet in Google search. (Mitchell et al. 2006)

Method 3: Entire sentences; use sentences in the web pages that contain the query to create a BOW vector.

We compared above methods in our Named Entity Recognition system (The details are discussed in section 3). The task is to recognize named entities for 5 classes -- *city, mammal, reptile, sport* and *religion.* Table 1 lists the precision (Eq. 1) of NER by using three methods to extract surrounding text individually. The approach to compute precision is illustrated in section 3.5.

Table 1. Performance Comparison of Different Methods to Create a BOW (using 40 web pages).

	Method 1	Method 2	Method 3
city	0.47	0.17	0.97
mammal	0.1	0.1	0.72
reptile	0.38	0.1	0.81
sport	0.29	1.0	0.65
religion	0.47	0.11	0.95
Avg.	0.34	0.29	**0.82**

Note that we used *40* web pages to create a BOW vector in all of three methods. This is an optimal number and we will discuss it later. In method *2*, the value of *k* is set to be the same as the snippet length used in Google Search. It is obvious that the Method *3*, which uses the entire query-contained sentences in the web pages, outperforms other methods. The entire web page introduced too much irrelevant information and therefore cannot precisely interpret named entities. The method *2* woks very well for class *sport* but works poorly for other classes. This implies that some classes may prefer close surrounded words and others need richer information to characterize named entities.

2. What is an appropriate number of web pages to create a BOW vector?

The hypothesis here is that creating a NLP item BOW by a certain small number of web pages can represent the probability distribution of a NLP item over entire web. We would like to answer how probable is it that this hypothesis is more accurate in general. Given our approach to create a BOW vector, the largest source of variability is the actual search results. A search engine can retrieve hundreds or thousands of top ranked web pages with respect to a query. Using fewer web pages to create a BOW vector leads to a less biased hypothesis. However, it may produce high variance. Statistically, we need to trade bias against variance by selecting an appropriate number of web pages to create a BOW vector. We launch an experiment as a task of Named Entity Recognition for 5 classes -- *city, mammal, reptile, sport* and *religion*. We evaluated their precisions using different number of web documents to create a BOW vector. Figure 2 lists NER precisions using BOW vectors generated from 10 to 200 web pages. For most of classes, the precisions stopped increasing at 40 or 60 pages. Considering the efficiency issue, we like to use 40 or 60 pages as an optimal web page number for all classes in our research.

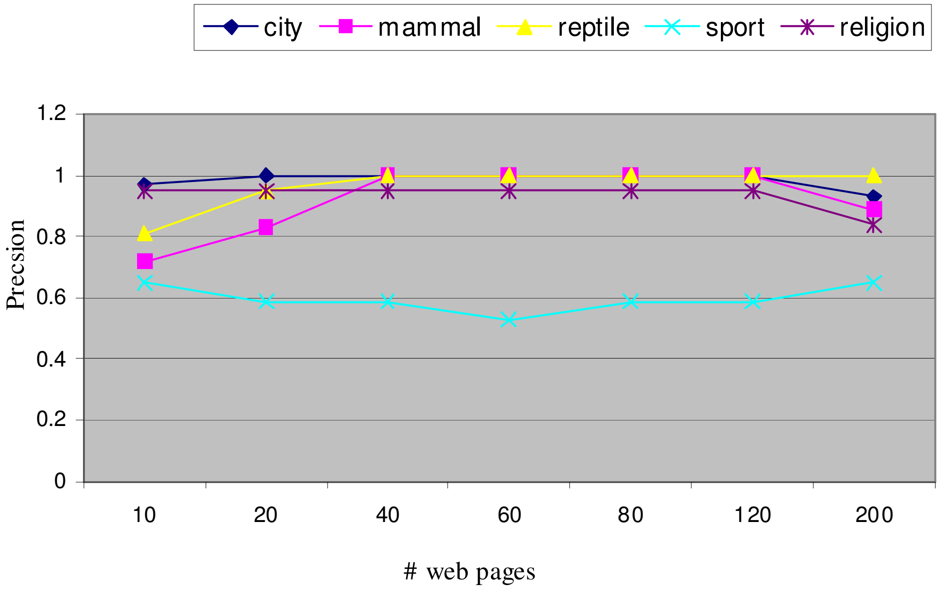


Figure 2. Precision Comparison of Different Number of Web Pages to Create a BOW

3. What is the advantage of generating BOW vectors from surrounding texts comparing with previous method?

(Collins and Singer, 1999) proposed to use spelling rules and context as features to characterize a named entity.

Table 2. Features used in Collins' system

full-string	*"Maury Cooper"*
contains(x)	*contains(Maury), contains(Cooper)*
allcap1	*A single word which is all capitals*
allcap2	*A single word which is all capitals or full periods*
nonalpha	*Any characters other than upper or lower case letters*
context_apposition	*apposition head, such as "vice president at S&P"*
context_prepersion	*A complement to a preposition, "work on"*
context-type	*context-type="appos" or "prep"*

The left column in Table 2 lists main features used by Collins' NER system; the right column gives brief illustrations. Collins' NER system applied these features on a family of supervised and unsupervised learning algorithms, and dealt with recognition of *person, location and organization*. To simplify our comparison with Collins' proposed features, we chose a data set from ACE (a data set widely used for named entity detection and annotation tasks.). This data set includes named entities from two classes, PER and GPE. PER stands for person names, and GPE is mostly location name. We used exactly the features listed in Table 2, and constructed BOW vectors to represent named entities of PER and GPE. We also generated BOW vectors using the method *3* as described in section 2.1. Then we applied two kinds of BOW vectors on a Naïve Bayes classifier individually, and evaluated their classification accuracy, which is how many training examples are correctly classified to PER and GPE. The average accuracy and standard derivation of 5-fold cross validation are listed as follows:

Table 3. Accuracy of using two kinds of BOW vectors on ACE data

Features	Accuracy
Collins' <spelling, context>	0.78 ±0.05
BOW by surrounding texts	0.95±0.02

The accuracy of using our BOW representation achieves 95%, which outperforms that using spelling and context features. It indicates that using surrounding texts returned by a search engine to construct a feature space is more relevant and informative. Our implementation of Collins' features achieves similar accuracy as the one in the paper (M. Collins and Y. Singer. 1999). We noticed that some spelling rules and contexts are not very significant features to characterize a class of named entities. For instance, some preposition contexts such as "part of" and "information about" appeared to be context features in both classes, which are not useful to distinguish one named entity from another. Again, it is hard to say that "allcap1" is a significant feature as well. Note that Collins' NER system also used co-training and boosting as their learning approaches. It would be an interesting direction for our future work.

Above, we have analyzed the feasibility and robustness of the BOW representation. Using Named Entity Recognition as an example, different proposed methods are evaluated and compared. We conclude that our BOW representation is sufficient and efficient to represent the probability distribution of words associated with a named entity on the web. The method of using surrounding sentences significantly outperformed other methods. Therefore, we like to use 40~60 top ranked web pages returned by a search engine and only use sentences containing the named entity to construct a BOW vector. In this sense, the named entity is characterized directly with their web contextual information, and contains a more refined and comprehensive coverage of various means in comparison to a fixed text corpus.

3 Named Entity Recognition

3.1 Related work

The goal of Named Entity Recognition (NER) is to extract named entities for a given class. Typical applications are to extract person names, locations and organization names from a fixed corpus.

Many research works on NER have been actively developed in recent years. (Yangarber et al, 2002) demonstrates the idea that learning several types of named entity simultaneously enables finding negative evidence (one type against all) and reduces over-generalization. (Cucerzan and Yarowsky, 1999) also use a similar technique and apply it to many languages.

(Collins and Singer, 1999) believe named entity patterns are important features and parse a complete corpus in search of named entity pattern candidates. They claim that both the spelling and the context of many named entity instances are sufficient to determine its type. They also present two learning approaches, co-training and boosting to recognize the named entities.

(Riloff and Jones, 1999) introduce mutual bootstrapping, which consists of growing a set of entities and a set of contexts in turn. The algorithm starts with a "seed list" of words that are known to be in-category, and then generates two resources: the "semantic lexicon" which is basically a wordlist for each category, and the "pattern dictionary", which is a list of patterns specifying contexts that usually appear with a given category. Riloff and Jones note that the performance of that algorithm can deteriorate rapidly when noise penetrates the entity list or pattern list. While they report relatively low precision and recall in their experiments, their work proved to be highly influential.

(Wang and Cohen, 2007) introduce a Google Sets style system, and propose a method for expanding sets of named entities in an unsupervised, domain and language independent fashion. Their system works by automatically finding semi-structured web pages that contain “lists” of items (normally the html lists) and then aggregating these “lists” so that the “most promising” items are ranked higher. They show that the system performs better than Google Sets in terms of mean average precision for the dataset tested.

3.2 Approach

Our approach is inspired by several previous works. Within our Read-the-Web framework, we predefined a knowledge base with multiple classes. Each class contains a few named entities. Another tactic is to predefine semantic relations among the classes. Figure 3 shows an example of a hierarchical structure of taxonomy in the knowledge base. For instance *"city"* is a class, this class is a subset of another class *"location"*, and mutually exclusive with other classes *"actor"*, *"company"*. Additionally, the class *"city"* contains seed named entities {"Boston", "Pittsburgh", "Paris"…}. In the following subsections, we will discuss how to take advantage of this information in the predefined knowledge base. For example, propagating seeds by using semantic relations among classes; using class name and seed names to find patterns, and therefore to extract candidate named entities by these patterns.

Table 4 gives pseudo code of our algorithm for Named Entity Recognition. Our approach is to learn several types of named entities simultaneously. We initialize a *seed sharing* process including sharing and propagating seeds among multiple classes. The idea of sharing seeds is to maximally obtain more labeled data for each class; we then generate a BOW vector for each seed using the method described in section 2.1, and train a classifier for each class using seed BOW vectors; next, we need to obtain candidate named entities, in other words, unlabeled data in a learning approach. The method is to use patterns as queries to retrieve web pages and extract candidate named entities. We also generate a BOW vector for each candidate named entity; we then classify and rank the candidate named entity by previous trained classifiers. A small number of top-ranked candidates are selected and added into their classes to re-train the classifiers. Ideally, we like to repeat step 3 and 4 until no more candidates can be selected. The evaluation of our algorithm is based on the precision. The recall is usually difficult to obtain. The details will be discussed in the next subsections.

Table 4. Algorithm for Named Entity Recognition

Start from a predefined knowledge base

1. Obtain more seeds by *seed sharing*;
2. Create a BOW vector for each seed,
3. *Train a classifier* using BOW vectors of seeds,
4. *Extract candidate named entities* from the Web
    - Derive heuristic patterns from the given class name.
    - Derive semantic patterns from the seeds
    - Use pattern as query to extract candidate named entities.
5. *Classifying and ranking candidate named entities* using the trained classifiers.
6. Ideally, keep doing steps 3 and 4 until no more candidates pass the classification.

3.2.1 Seed Sharing

The idea here is to maximize labeled data for each class, and therefore to obtain more labeled data and benefit for the training process. As introduced earlier, figure 3 shows an example of a predefined hierarchical structure of taxonomy in the knowledge base. The semantic relations we used are “subset” and “exclusive”. We build a tree using the “subset” relation among classes, and explicitly define “exclusive” relation among classes. For example, *“company”* and *“university”* are subsets of a class *“organization”*; *“bank”* and *“IT_Company”* are subsets of *“company”* etc. A small number of labeled data (positive seeds) has been assigned for these classes. We are now interested in increasing the number of labeled data by propagating the labeled data within each class into other classes by using known semantic relations.

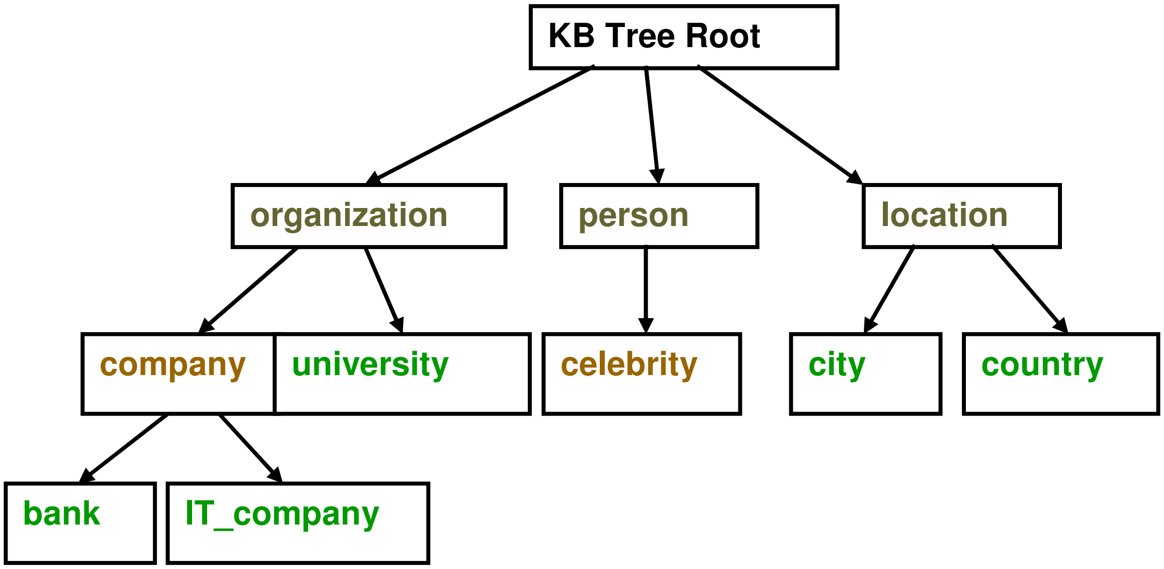


Figure 3. An illustration of taxonomy

We can formally define rules for *seed sharing*. Here seed means the positive and negative labeled data. We denote A_+ as the positive seed set of a class A and A_- as the negative seed set. The seed sharing rules can then be defined as the following:

Rule 1: If the class A is ascendant of the class B, then

$$A_+ \leftarrow A_+ \cup B_+$$
$$B_- \leftarrow A_- \cup B_-$$

Rule 2: If the class A is mutually exclusive of the class B, then

$$A_- \leftarrow A_- \cup B_+$$
$$B_- \leftarrow A_+ \cup B_-$$

In this sense, given a class *“company”* as an example, the positive seeds of *“bank”* and *“IT_company”* can be positive seeds of *“company”* by the “subset” relation. Also the positive seeds of *“university”*, *“location”* and *“person”* can be negative seeds of *“company”* by the “exclusive” relation. Figure 4 shows a performance improvement for a *"company"* classifier as more seeds are added by rules. When x-axis is 0, we only used initial seeds of the class *"company"* to train a classifier, the classification accuracy is 76%; then in step 1, the class *"company"* shared more seeds from *"IT_company"* by "subset *IT_company company"*, we retrained the classifier, and the classification accuracy increased to 80%; again in step 2, the class *"company"* obtained more seeds from *"University"* as its negative seeds by "exclusive *company university"*, we retrained the classifier, the classification accuracy increased to 82%; Finally, after adding more seeds by using all the semantic relations related with the class *"company"* , we retrained the classifier, the classification accuracy increased to 88%. This increasing trend proves the advantage of our *seed sharing* approach.

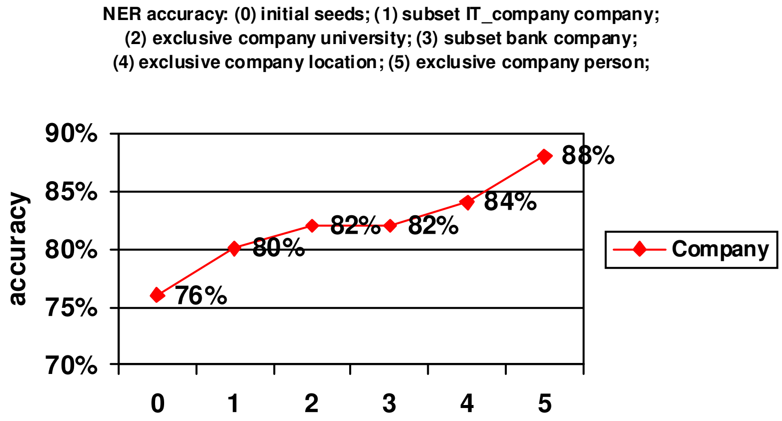


Figure 4. Performance Change of the "*Company"* Classifier as the Procedure of *Seed Sharing*

3.2.2 Train Classifier

After the process of *seed sharing*, we train a classifier for each class using positive and negative labeled data. The approach is to generate a BOW vector for each seed named entity. We then simply used these BOW vectors to train a Naïve Bayes classifier or a Logistic Regression classifier.

Naïve Bayes classifier is a highly practical Bayesian learning method and widely applied in natural language text documents. It is based on the simplifying assumption that attribute values $X(x_1, x_2, \ldots, x_n)$ are conditionally independent given a target class *Y*. Logistic Regression leads to a linear classification rule and directly estimates the parameters of *P(Y|X)*, whereas Naive Bayes directly estimates parameters from *P(Y)* and *P(X|Y)* (Wasserman, 2004). The Naïve Bayes and Logistic Regression converge toward their asymptotic accuracies at different rates. In many case, the Naïve Bayes outperforms Logistic regression when training data is scarce.

3.2.3 Extracting Candidates from the web

To obtain candidate named entities, in other words, unlabeled data in a learning approach, we used patterns as queries to retrieve them from the web directly. The patterns are defined as left or right context phrases for named entities. There are two types of patterns:

heuristic patterns derived from the class names and semantic patterns from seeds. We then used these patterns as search queries to retrieve several web pages that contain candidate named entities one the web. Several language techniques were used to extract formatted candidate named entities.

3.2.3.1 Derive heuristic patterns from class names

The predefined knowledge base includes multiple classes. For each class, we derived two heuristic patterns "*Y* such as" and "*Y* including" (Hearst, 1992). *Y* is a plural of class name. For instance, if a class name is "city", then heuristic patterns are "*cities such as*" and "*cities including*".

3.2.3.2 Derive semantic patterns from seeds

Semantic patterns of seeds are also good queries to retrieve candidates. Using a seed as a query in a search engine to retrieve a number of top-ranked web pages, we then segmented web documents into sentences and only tagged part-of-speech on sentences that containing seed named entities. Inspired by context features used in Table 2 (Collins and Singer, 1999), we applied two grammar rules on sentences to extract semantic patterns:

Rule 1: (Verb/Nouns + Preposition) +NE
Rule 2: NE+ (auxiliary Verbs + Nouns)

For instance, "ceo of" meets the need of (Verb + Preposition), and "is a company" matches the form (auxiliary Verb + Noun). NE means seed named entities. Table 5 lists examples of extracted semantic patterns. We can see some semantic patterns are very significant to distinguish a class of named entities. For example, "mayor of" is clearly a pattern of class "*city";* "biography of" is a pattern of class "*people*".

Table 5. Semantic patterns examples

works of _, writings of _, life of _, quotes by _ genius of _, biography of _ , works by_, collection of _, portrait of _ stories of _, behalf of _, ceo of _, profile for _, president of _, subsidiary of _, agreement with _, directors of _, owned by _, filed under _, reports that _ hotels in _, hotel in _, mayor of _, city of _, forecast for _, map of _, located in _, north of _, government of _ , map of _, cities in _, languages of _, , anthem of _, destinations in _, culture of _, business in _, population of _, research the new _, alternator on _, array of _, colors available for _, specials on _, price on _, quote for _, ……

3.2.3.3 Extract candidate named entities using patterns

We used derived patterns as queries into a search engine and retrieved a number of top-ranked web pages. We are interested in sentences in these web pages such as, "cities such as Boston, Paris and Beijing…" or "mayor of New York". Notice that a named entity in a sentence is usually either a subject following by a verb or an object connecting to a preposition." we therefore extracted named entities from sentences that satisfy at least one of the following two rules:

Rule 1: NE (*subject*) + Verb
Rule 2: a Preposition + an NE (*object*)

The extracted token strings may be noisy. We removed stop words as well. In this way, we get quite cleaned named entities. The number of extracted named entities is varying in different classes because it depends on how many sentences that patterns can match with.

3.2.4 Classification and Ranking

The candidates named entities are ranked for each class using the classifier that trained previously. This procedure is classification and our ranking function is a log-likelihood ratio between positive class and negative classes. The detailed equation is as follows.

$$score(X) = \log(P(X \mid Y = 1)) - \log(P(X \mid Y = 0))$$

Here *X* means a named entity BOW vector, and *Y* represents a target class. *Y=1* means positive class, and *Y=0* means negative class.

Table 6 gives an example ranking list for a class "*city*". Candidate named entities are sorted by their score of the log-likelihood ratio. Top-ranked candidate named entities excluding seeds will be kept as our result of Named Entity Recognition.

Table 6. Ranking list for class "*city*"

Boston = 28381.445
Atlanta = 21498.790
Seattle = 18676.790
Baltimore = 17197.822
Newark = 16169.043
Shanghai = 15073.673
San Francisco = 14549.210
Chicago = 14542.440
Detroit = 13468.306
Schenectady = 12296.788
Cleveland = 12181.007
Cincinnati = 12062.138
Manchester = 11487.970
……

3.3 Results and Evaluation for Named Entity Recognition

We predefined a knowledge base that contains more than 20 classes. Each class has about 15 positive seeds, which referred from (D. Nadeau. 2007). Table 7 lists results of top-ranked named entities for the 20 classes. Seed named entities are all excluded in the ranking list of candidate named entities.

Table 7. Top-ranked NEs that extracted from the web.

Class	**Examples of top-ranked named entities**
actor	Gene Hackman, Pierce Brosnan, George Clooney, Bruce Willis, Kate Winslet, Marisa Tomei, Jamie Foxx, Johnny Depp, Meryl Streep, Liev Schreiber, Morgan Freeman, …
amusement park	Disney World, Coney Island, Magic Kingdom, Splash Mountain, Dorney Park, San Feliu, Wildwater Kingdom, Hurricane Harbor, Typhoon Lagoon, Cedar Point,

	Disney parks, …
bank	Deutsche Bank, Bank India, Julius Baer, Goldman Sachs, JPMorgan Chase, ABN AMRO, Wells Fargo, Merrill Lynch, Barclays Bank, Bear Stearns, ICICI Bank, …
car	Ford Focus, Toyota Prius, Civic Hybrid, Elantra Touring, Toyota, PT Cruiser, Lexus, Corolla, Honda Civic, Honda Accord, Spectra5, …
celebrity	James Denton, Greta Garbo, Tom Watson, Joan Crawford, Tom Hanks, Clark Gable, Marlene Dietrich, Kerry Katona, Paris Hilton, Dustin Hoffman, …
city	Madrid, London Gatwick, London, York City, Vancouver, Los Angeles, Des Moines, Buenos Aires, Tokyo, San Francisco, Edinburgh, …
company	Vonage Holdings, NYSE Euronext, British Airways, Philips Electronics, American Express, American Airlines, Merrill Lynch, Vitesse Semiconductor, Adelphia Communications, Tumbleweed Communications, Washington Mutual, …
disease	Lyme disease, rheumatoid arthritis, ulcerative colitis, rheumatic fever, liver cancer, ovarian cancer, Yellow fever, yellow fever, Adjustment Disorder, Japanese encephalitis, autoimmune hepatitis, …
drug	respiratory failure, sleeping pills, Anaprox DS, antiplatelet drugs, potential effects, Adderall XR, anti-seizure medications, alcohol dependence, Metadate ER, MAO inhibitors, generic Rx, …
food	raw milk, Palm Oil, soy sauce, canola oil, wheat flour, brown rice, vitamin B12, ice cream, green beans, Brussels sprouts, cottonseed oil, …
insect	beetles, termites, flies, spiders, ants, aphids, mites, cockroaches, wasps, fleas, trichogramma, …
mammal	deer, mule deer, whales, horses, dogs, grizzly bear, elephants, deer mouse, voles, mice, brown bear, …
mineral	folic acid, calcium carbonate, zinc oxide, specific gravity, zeta potential, iron ore, iron oxides, surface hydrophobicity, tap water, barrier layer, synthetic vitamins, …
movie	Indiana Jones, Donnie Darko, Van Helsing, harry potter, Mortal Kombat, Miss Pettigrew, Karate Kid, Hidden Dragon, Resident Evil, Dark Knight, Tomb Raider, …
nation	Saudi Arabia, Czech Republic, Equatorial Guinea, United Kingdom, Sri Lanka, East Timor, San Marino, European Union, Ottoman Empire, climate change, including Syria, …
newspaper	Chicago Sun-Times, Le Figaro, Glenview Announcements, Straits Times, La Presse, York Post, Boston Globe, El Republicano, USA TODAY, Le Droit, El Federalista, …
religion	Judaism, Buddhism, Islam, Shinto, Hinduism, Christian Science, God, Muslims, Christianity, Taoism, revelation, …
reptile	iguana, snakes, iguanas, turtles, lizards, Bearded dragons, bearded dragons, tortoises, Tortoises, desert tortoise, caiman, …
sport	golf, volleyball, boxing, wrestling, soccer, basketball, Baseball, swimming, ice hockey, disc golf, table tennis, …
university	Pepperdine University, University Chicago, undergraduate education, University Tokyo, universities Students, Louvain UCL, University Essex, University Windsor, University Maryland, Mons FPMs, University Birmingham, …

We can see that most of the top-ranked named entities correctly belong to target classes. Typically, precision and recall are widely used to statistically evaluate information retrieval results. Since recall is hard to obtain in our case, it is unlikely to get entire named entities for most of given classes. We only consider precision in the top *N* named entities of the ranking list as our evaluation metric,

$$precision = \frac{\# correct \quad NEs}{N} \qquad (1)$$

Figure 5 illustrates precision in the top 10, 20, and 30 of the ranking list. The average precision over 20 classes obtained **96%, 93%** and **90%** for top 10, 20 and 30 respectively. Note that the named entities in the ranking list should have a positive score. This is more strict evaluation comparing only check the ranking itself. We also consider the test of significance. The stability of our results greatly relies on the Google search results. We examined the experiment results during a period of one semester, and found slight variability. The resulting p-value is less than 0.01. Therefore, we have a strong significant confidence for our experiment results.

We found errors such as "university students" in class "*university*", "climate change" in class "*nation*" etc. This kind of named entity is truly relevant with the target class. In more detail, the word distribution of "university students" is similar to some other word distributions of university named entities. The "*university*" classifier therefore predicted "university students" as a true university name. To address this problem, we like to take advantage of patterns which we described in subsection 3.2.3. From class names and seeds, each class can obtain a series of derived patterns. We believe that true university named entities should frequently co-occur with these derived patterns on the web. For instance, "University Chicago" instead of "university students", is likely to appear with derived patterns "research on _" or "university such as". This would be our future research work.

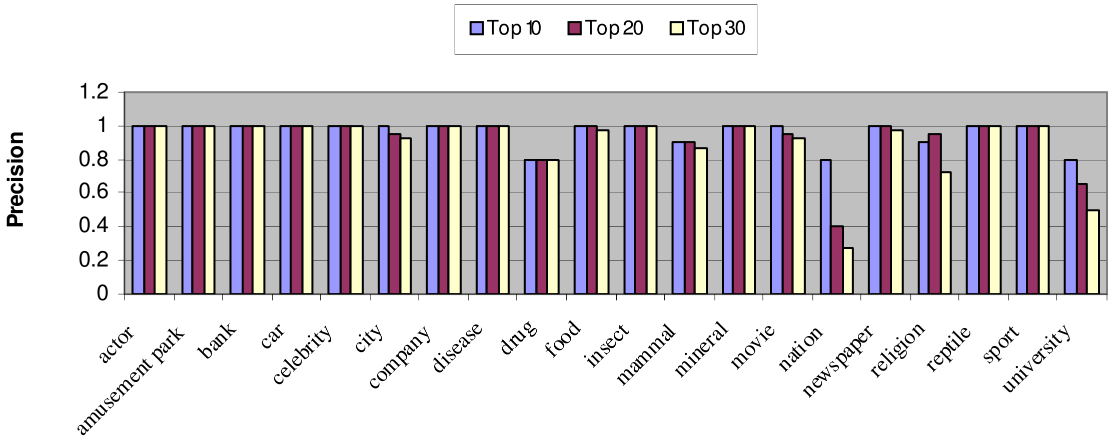


Figure 5. Precisions of top 10, 20, and 30 ranked named entities from the web.

3.4 Summary of Named Entity Recognition

We have been discussed our implementation of Named Entity Recognition. Using a predefined knowledge base, our approach performed multi-class learning simultaneously and obtained more labeled data by seed sharing. The BOW representation characterizes word distributions of named entities directly from web. The results demonstrate that we can attain high precisions for more than 20 classes in the experiment.

4 Relation Extraction

4.1 Related work

Entity relation extraction is a task that finds predefined relations between entities. Entities are individuals of selected semantic elements, for example person, location, and organization. For instance, “Bill Gates works at Microsoft Inc.” represents a person-affiliation (person, company) relationship. The goal of relation extraction is to recognize tuples (Bill Gates::Microsoft Inc) that satisfy particular relations such as person-affiliation relationship from unstructured text. Other examples of relations could be wasBornIn(person, location), organization-location(company, location) etc. Many previous applications of entity relation extraction have been widely used in question-answering, semantic search, information extraction, text summarization, language modeling etc. We particularly introduce some approaches which are tightly closed to our work.

(Brin 1998) presented a system called DIPRE (Dual Iterative Pattern Relation Extraction) which generates lists of book titles paired with book authors. The relation of interest to DIPRE is the (author, book). DIPRE started with a small set of (author, book) pairs, then used lexical features implemented by regular expressions to extract a tuple for every instance of a (author, book) seed pair in relative proximity. For example, the tuple extracted for (*Shakespeare, King Lear*) for the string, “Consider Shakespeare's play King Lear, which tells the tale ...” Each extracted tuple is then grouped to induce a pattern: The longest common suffix of the left field and the longest common prefix of the right field were extracted. Using such a pattern allows the system to extract new examples of (author, book) pairs. In turn these pairs can generate new patterns. One of the central insights of DIPRE is that the size of the web allows the use of extremely selective patterns to induce new example pairs of (author, book). Even with extremely selective patterns, new seed examples will be introduced due to the sheer size of the web. Hence, DIPRE explicitly maintains selectivity by using highly precise patterns and implicitly increases coverage through the size of the unlabeled data set.

Snowball (Agichtein and Gravano. 2000) is essentially an improved version DIPRE and shares much of DIPRE's architecture. It presents the system Snowball for extracting relations from unstructured text. Snowball shares much in common with DIPRE, including the employment of the Yarowsky bootstrapping framework () as well as the use of pattern matching to extract new candidate relations. The relation that Snowball focuses on is the (organization, location) relation.

The primary advantage of Snowball and other related semi-supervised training systems is that they require little to no human annotation. However, Snowball relies on an intrinsic property of organizations and locations that every organization has its headquarters in only one location - when calculating the confidence score of a pattern. This property does not hold for all relations. For instance, in the author-of relation, one author can be associated with many books and one book with many authors. Even organizations can have multiple headquarters in different parts of the world.

(Banko et. al., 2007) present an unsupervised framework for relation extraction directly from Web. They introduced Open IE (OIE), a new extraction paradigm where the system makes a single data-driven pass over its corpus and extracts a large set of relational tuples without requiring any human input. They also presented TEXTRUNNER, a fully implemented, highly scalable OIE system where the tuples are assigned a probability and indexed to support efficient extraction and exploration via user queries. The key idea in their approach is to automatically label training data for the relation classifier via linguistic constraints imposed by a parser. Unlike the kernel methods which require labeled set of training instances, this approach can generate its own training data.

4.2 Approach

Our approach still used the idea of BOW representation. We believe that tuples, for example, {*(Elvis Presley:: Tupelo), (Albert Einstein :: Ulm), (George Washington :: Westmoreland Country)*}, which represent relationship "*wasBornIn*" should have similar word distribution on the web. We were also inspired from Richard Wang's SEAL system (Wang and Cohen, 2007). They introduced a Google Sets style system to expand sets of named entities by automatically finding semi-structured web pages that contain "lists" of items (normally the html lists) and then aggregating these "lists" so that the "most promising" items are ranked higher. We extended their method to find "lists" that contains relational tuples instead of named entities.

Table 8 gives pseudo code of our algorithm for Relation Extraction. A few of seed relational tuples were given at first. We can use relation "*wasBornIn*" as an example. A BOW vector was generated for each seed tuple. The details will be discussed in section 4.2.1; we then trained a classifier using seed BOW vectors. Candidate relational tuples were extracted by using a wrapper extraction (Wang and Cohen, 2007) based on HTML patterns. The detailed method will be described in section 4.2.2. We also generated a BOW vector for each candidate relational tuple; we then classified and ranked the candidate relational tuples by previous trained classifiers. A small number of top-ranked candidates are evaluated. We still used precision (Eq. 1) as our evaluation metric.

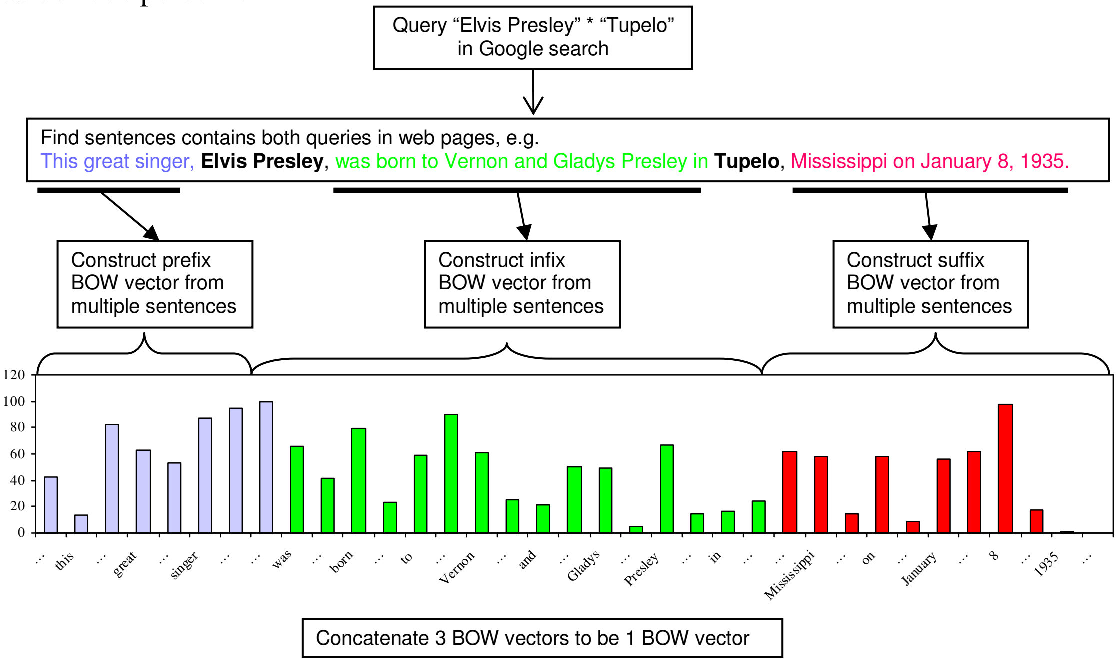
Table 8. Algorithm for Relation Extraction

```
1. Create BOW vectors for each seed relational tuple;
     a. Each seed contains two named entities, such as
        <Elvis Presley :: Tupelo>.
2. Train a classifier using seed BOW vectors;
3. Use a wrapper extraction based on HTML patterns
   (Wang and Cohen, 2007) to extract candidate relational
   tuples from the web;
     a. Create BOW vector for each candidate.
4. Classifying and ranking the candidates by their BOW.
```

Step 2 and 4 are similar to our learning approach to Named Entity Recognition. We will be focus on descriptions of step 1 and 3 in the following subsections.

4.2.1 BOW representation for relational tuple

In subsection 2.1, we demonstrated how to construct a BOW vector for a named entity. Based on this, we have a particular method to represent a relational tuple, a pair of named entity in our case. For instance, given a tuple "<*Elvis Presley:: Tupelo*>", we typed it into a search engine, and then used our language engineering method to retrieve sentences that contain the query; we collected surrounding words of prefix, that is words before "Elvis Presley", words between "Elvis Presley" and "Tupelo" as infix and surrounding words after "Tupelo" as suffix. We then generated three BOW vectors for prefix, infix and suffix individually, and concatenate three BOW vectors as a single one BOW vector to represent the relational tuple "<*Elvis Presley:: Tupelo* >". The figure below illustrates how to construct a BOW vector of "<*Elvis Presley:: Tupelo*>". The x-axis represents the word tokens appeared in the prefix, infix and suffix respectively, and the y-axis is the counting frequency of tokens that co-occur with the query "Elvis Presley, Tupelo". The advantage of this particular representation is to capture order information of a relational tuple. For this example, it is "person *was born in* location", not confusing with "location *was born in* person".


Query "Elvis Presley" * "Tupelo" in Google search
Construct prefix BOW vector from multiple sentences
Construct infix BOW vector from multiple sentences
Construct suffix BOW vector from multiple sentences
120
100
80
60
40
20
0
…
this
…
great
…
singer
…
…
was
…
born
…
to
…
Vernon
…
and
…
Gladys
…
Presley
…
in
…
…
Mississippi
…
on
…
January
…
8
…
1935
…
Concatenate 3 BOW vectors to be 1 BOW vector

Find sentences contains both queries in web pages, e.g.
This great singer, **Elvis Presley**, was born to Vernon and Gladys Presley in **Tupelo**, Mississippi on January 8, 1935.

Figure 6. The illustration of a BOW vector of *"<Elvis Presley:: Tupelo>"*

4.2.2 Extract candidates from HTML sources text

Different from our approach to extract candidates for Named Entity Recognition, we used a wrapper extraction (Wang and Cohen, 2007) based on HTML patterns to obtain candidate relational tuples. Their wrapper extraction assumes that entities belonging to the same class will be linked by appearing in similar formatting structures on the same web page. We replaced their entities to be relational tuples, and used similar strategy to construct HTML patterns. For instance, we were given three seed relational tuples as follows,

```
Benjamin Franklin :: lightning rod
```

```
Alexander Graham Bell :: telephone
  Thomas Edison :: Phonograph
```

We then typed these seeds as a query to Google Search, and extracted a number of web pages which contain all seeds. Similar information in semi-structured web documents can be formatted quite differently on different pages, but fairly consistently within a single page. For above example, we found a pair of inventor and invention names is in a list which is embedded with "<tr><td>" (to the left) and "</td></tr>" (to the right) in one web page. The table below illustrated a HTML source text that contains pairs of inventor and invention names.

Table 9. HTML sources text contains "invention of" tuples
HTML sources text from "montgomeryschoolsmd.org" ("…" is omitted text).

```
  …  <tr>
      <td><div align="center"><font face="Georgia, Times New Roman,
Times, serif">1837</font></div></td>
      <td><div align="center"><font face="Georgia, Times New Roman,
Times, serif">John Deere</font></div></td>
      <td><div align="center"><font face="Georgia, Times New Roman,
Times, serif">Steel Plow</font></div></td>
    </tr>
    <tr>
    <td><div align="center"><font size="3" face="Georgia, Times New
Roman, Times, serif">1876</font></div></td>
      <td><div align="center"><font size="3" face="Georgia, Times New
Roman, Times, serif">Alexander Graham Bell</font></div></td>
      <td><div align="center"><font size="3" face="Georgia, Times New
Roman, Times, serif">Telephone</font></div></td>
    </tr>
    <tr>
      <td><div align="center"><font size="3" face="Georgia, Times New
Roman, Times, serif">1877</font></div></td>
      <td><div align="center"><font size="3" face="Georgia, Times New
Roman, Times, serif">Thomas Edison</font></div></td>
      <td><div align="center"><font size="3" face="Georgia, Times New
Roman, Times, serif">Phonograph</font></div></td>
    </tr>…
```

We used a set of HTML patterns suggested in (Wang and Cohen, 2007) to exact candidate relational tuples. Recall that we used heuristic patterns and semantic patterns for Named Entity Recognition. The heuristic patterns "*Y* such as" and "*Y* including" usually do not applicable for a relational tuple; the grammar for semantic patterns needs to be more complicated to apply a pair of named entities. Also only considering an infix semantic pattern could not be general for many relational tuples, since many infix between two named entities are stop words or no semantic words. Using a HTML pattern first avoided costs on language engineering. Second, majority of extracted relational tuples were quite clean without noisy tokens.

Each candidate was created a BOW vector using the method described in section 4.2.1. We then applied trained classifier to predict candidate relational tuples. A ranking list was sorted based on candidates' classification score.

4.3 Results and Evaluation for Relation Extraction

We applied our method on six widely referred relations -- "*partnership*", "*CEOof*", "*starIn*", "*inventionOf*", "*companyLocation*" and "*wasBornIn*". We provided 2~3 seeds for each relation. The candidate extraction can produce 40~130 candidate relational tuples for six different relations. Table 10 listed the top-ranked relational tuples for these six relations. Seeds are all excluded in the ranking list.

Table 10. Relation Extraction results

Seeds	**Harrison Ford :: Calista Flockhart** **Brad Pitt :: Angelina Jolie**	**Bill Gates :: Microsoft** **Larry Page :: Google** **Jerry Yang :: Yahoo**
Top ranked candidates	Heath ledger :: Michelle Williams	Steve Chen :: YouTube
	Tony parker :: Eva Longoria	Sergey Brin :: Google
	Tom cruise :: Katie Holmes	Steve Wozniak :: Apple
	Matthew Broderick :: Sarah Jessica Parker	Steve Ballmer :: Microsoft
	John Travolta :: Kelly Preston	David Filo :: Yahoo
	Macaulay Culkin :: Mila Kunis	Max Levchin :: Paypal
	Justin Timberlake :: Jessica Biel	Scott Mcnealy :: Sun Microsystems
	Seal :: Heidi Klum	Tom Anderson :: Myspace
	Keith Urban :: Nicole Kidman	Chad Hurley :: Youtube
	Vince Vaughn :: Jennifer Aniston	Philip Knight :: Nike
	Warren Beatty :: Annette Bening	Gordon Bowker :: Starbucks
	Gavin Rossdale :: Gwen Stefani	Jim Casey :: UPS
	Jesse James :: Sandra Bullock	Bernie Marcus :: Home Depot
	Nick Lachey :: Jessica Simpson	Mark Cuban :: Icerocket
	...	...
Seeds	**Julie Andrews :: The Sound of Music** **Keanu Reeves :: The Matrix** **Nicolas Cage :: National Treasure**	**Apple :: Cupertino** **Google :: Mountain View** **Microsoft :: Redmond**
Top ranked candidates	Clint Eastwood:: Dirty Harry	Visa:: Foster City
	Jennifer Lopez :: El Cantante	Vontu:: San Francisco
	Warren Beatty:: Shampoo	Wind River Systems:: Alameda
	Joan Crawford:: Mildred Pierce	Apple:: Santa Clara
	Sean Connery:: The Man Who Would Be King	Symantec:: Santa Monica
		Coremetrics:: San Mateo
	Angelina Jolie:: Girl, Interrupted	LookSmart:: San Francisco
	Dustin Hoffman:: The Graduate	Consorte Media:: San Francisco
	John Wayne:: True Grit	Oracle:: Redwood City
	Bing Crosby:: Going My Way	Ecast:: San Francisco
	Bruce Willis:: Die Hard	Avvenu:: Palo Alto
	The Marx Brothers:: Duck Soup	Efficient Frontier:: Mountain View
	Uma Thurman:: Dangerous Liaisons	Adobe Systems:: San Jose
	Goldie Hawn:: Cactus Flower	Yipes Enterprise Services:: San Francisco
	Doris Day:: Send Me No Flowers	
	Woody Allen:: Annie Hall	Baynote:: Cupertino
	Meryl Streep :: Mamma Mia	Hewlett Packard:: Palo Alto
	Jackie Chan:: Rush Hour	
	Adam Sandler:: Happy Gilmore	......
	......	

Seeds	**Benjamin Franklin :: lightning rod** **Alexander Graham Bell :: telephone** **Thomas Edison :: Phonograph**	**Albert Einstein :: Ulm** **Elvis Presley :: Tupelo** **George Washington :: Westmoreland County**
Top ranked candidates	Thomas Edison:: Light Bulb Herman Hollerith :: punch-card tabulation machine system Jan Matzeliger:: Shoe-Lasting Machine Henry Bessemer:: Bessemer Steel Process William Bullock:: Web-Perfecting Process John Pulitzer:: Newspapers Norbert Rillieux:: Refining Sugar Elijah McCoy:: Lubricating Cup Eli Whitney :: cotton gin Edwin Drake:: Oil Refinery William Burroughs:: Adding Machine George Pullman:: Pullman Car Johannes Gutenberg :: press Christopher Sholes:: Typewriter Washington Roebling:: Brooklyn Bridge John Deere:: Steel Plow	Amelia Earhart:: Atchison Allan Border:: Sydney Anna Pavlova:: St. Petersburg Antonin Dvorak:: Nelahozeves Antonio Gaudi:: Reus Aristotle Socrates Onassis:: Smyrna Anthony Burgess:: Harpurhey Barry Gibb:: Douglas Bruce Willis:: Idar-Oberstein Alfred Bernhard Nobel:: Stockholm Antonio Vivaldi:: Venice Boutros Boutros-Ghali:: Cairo Agatha Christie:: Torquay Carl Lewis:: Birmingham Anne Frank:: Frankfurt Abraham Lincoln:: Hodgenville Adolf Hitler:: Branau Andrew Carnegie:: Dunfermline Andy Warhol:: Pittsburgh ....

Most of the results accurately belong to their corresponding relations. We used precision (Eq. 1) in the top *N* of the ranking list as our evaluation metric. The precision of top 10, top 20 and top 30 are calculated and shown in Figure 7.

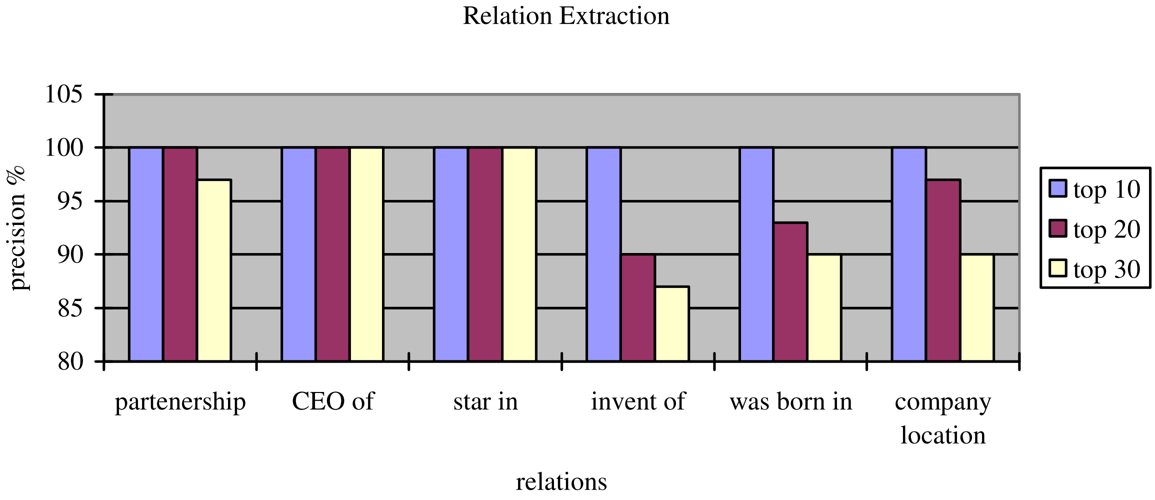


Figure 7. Performance of Relation Extraction

Figure 7 indicated that our method can achieve a high precision, which implied that the advantage of candidate extraction is to greatly avoid noise data comparing with the extraction results using the semantic patterns. Moreover, our particular BOW representation correctly captured the word distribution of relational tuples on the web. In this sense, our classifiers resulted in accurate predictions.

Our future work is to apply our method on more relations, and perform a Bootstrapper as suggested in (Wang and Cohen, 2008).

4.4 Summary of Relation Extraction

From above discussion of our implementation of Relation Extraction, we can see that the BOW representation for relational tuple does capture the order information of arguments in the binary relation. The strategy to extract “lists” of relational tuples on semi-structured web pages by HTML based patterns is efficient and accurate. Our learning approach is weakly-supervised with only 2~3 seeds as input. The results demonstrate that we can attain high precisions in 6 well-known relations.

5 Prepositional Phrase Attachment

5.1 Related work

Prepositional Phrase Attachment (PPA) is a common cause of structural ambiguity in natural language processing. For example, take the following sentence:

“The girl saw the butterfly with the telescope.”

The prepositional phrase 'with the telescope' can either attach to the subject NP 'the girl' or to the object NP 'the butterfly', giving two alternative structures. (In this case the subject NP attachment is correct):
Subject NP-attach: (the girl (with the telescope))
Object NP-attach: ((the butterfly (with the telescope))

Prepositional phrases, such as “with the telescope” is ambiguous. We know its correct attachment by our background knowledge. Otherwise, it is hard to determine immediately whether the girl was with the telescope and saw the butterfly, or whether the girl saw the butterfly that with the telescope. Such an undertaking is mostly trivial to humans, while it remains elusive for a computer system to find an algorithmic solution.

The prepositional phrase attachment is necessary for the understanding of text and even for the information extraction because they affect the meaning of sentences in a critical manner. However, an efficient disambiguation of prepositional phrase attachments is a very difficult problem in natural language processing (Brill and Resnik, 1994) (Harabagiu, 2000). Difficulties arising from this task are rooted in the fact that the problem encompasses not only lexical ambiguities but also semantic and thematic ambiguities.

Lexical information is provided by the sentence structure and contains not only part-of-speech information but also sentence syntax. This kind of information may not always be enough to disambiguate prepositional phrase attachments, but it is a valuable asset in narrowing down the scope of the problem by delineating the valid attachment forms. Semantic information determines the sense of the words involved in a prepositional attachment, which can help us in fixating the type of prepositional phrase attachment we have in a manner similar to the lexical information. Thematic information is always contextual information of the sentence. Moreover, domain knowledge plays an important role on determining static prepositional phrase. For instance, proper nouns such as “Bank of Commerce”, its attributed sense may not be derived from the individual senses of its components. Particular domain knowledge (business journals, for example) may provide important information on which kinds of phrases have a particular sense and attachment type differing from the one that would be determined by the independent senses of their components.

Numerous approaches have been proposed and used the above four types of information to disambiguate prepositional phrase attachments. In recent years the focus in natural language processing in general has shifted towards statistical methods. We shall review several approaches as follows.

(Hindle and Rooth, 1991) proposed that utilized distributional frequencies within an automatically parsed corpus to determine the relative associative strength between a preposition and verb and noun phrase heads. The method was based on lexical association and therefore relies purely on syntactic knowledge. The training set is composed of triples of the form [V P, NN, PP]. For each triple, the number of times it occurs within a text is counted, providing the frequency information needed for the next step. Once all the distributional frequencies have been collected, disambiguation is a matter of calculating the relative strength of association between a preposition and the verb phrase head versus the likelihood of said preposition being attached to the noun phrase head, and selecting the better one. The efficiency of this method is in the 80% range.

The statistical back-off model, pioneered by (Collins and Brooks, 1995), is a more advanced approach to statistical prepositional phrase attachment disambiguation. It works by calculating the frequencies for [V P, NP1, Prep, NP2] 4-tuples based on the frequencies of previous attachments with the same phrase heads. In this sense it is similar to the corpus-based statistical disambiguation algorithm. It differences itself from the former, however, in that it utilizes an approach that mimics the backed-off n-gram model in how it handles data sparsity. This algorithm falls back to triples and eventually word pairs if no frequencies for similar 4-tuples can be found. Of all purely statistical methods, this one performs best at a precision of 84.5%.

(Pantel, 2000) presents an unsupervised approach that achieves similar performance to supervised methods. Unlike previous unsupervised approaches in which training data is obtained by heuristic extraction of unambiguous examples from a corpus, they used an iterative process to extract training data from an automatically parsed corpus. Attachment decisions are made using liner combination of features and low frequency events are approximated using contextually similar words.

5.2 Approach

The previous approaches usually used lexical, semantic, thematic and domain knowledge as features for a learning procedure. These types of features usually relied on parsing, grammar rules or corpus scanning to obtain, which is complicated and costly. We proposed our BOW representation as a global feature to help disambiguate the attachment of prepositional phrase.

Table 11 gives pseudo code of our algorithm for Prepositional Phrase Attachment. Given a sentence with a prepositional phrase, a subject and an object, for instance,

> “***Mary ate the salad with croutons.***”

We first extracted a named entity (NE) "Mary" in subject, a named entity "salad" in object and a named entity "croutons" in the prepositional phrase (PP). The goal is to determine the prepositional phrase "with croutons" is an attachment of "Mary" or it is an attachment of "salad". Second, we created BOW vectors for these three named entities using the method as described in section 2.1. Third, we computed KL-divergences between (BOW(croutons), BOW(Mary)) and (BOW(croutons), BOW(salad)). Here the KL-divergence was treated as a distance function to measure the distance between word distributions of “croutons” and “Mary”, again “croutons” and “salad” individually. Finally, we compared two KL-divergence values. The prepositional phrase was determined to attach with the named entity with lower KL-divergence value.

Table 11. Algorithm for Prepositional Phrase Attachment

```
1. In each sentence with a prepositional phrase, extract
   three named entities:
       e. g. “Mary ate the salad with croutons.”
       a. subject named entity – “Mary”
       b. object named entity – “salad”
       c. the named entity in the prepositional phrase –
          “croutons”
2. Create BOW vectors for each of them.
3. Compute the KL-divergence between
       a. <NE in PP, NE in the subject>
       b. <NE in PP, NE in the object>
4. Compare two KL-divergence values. PP attached to the
   NE with lower KL-divergence value.
```

Our assumption here is that the word distribution of the “croutons” should be more similar to the word distribution of the “salad” than to the word distribution of a person “Mary”. More generally, we assume the question of whether the prepositional phrase should attach to the subject or the object can be answered by which one has a more similar BOW distribution to that of the entity in the prepositional phrase. The detailed KL-divergence equation is as follows,

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} . \tag{2}$$

We try to measure the difference between two probability distributions *P* and *Q*. In our case, *P* represents the BOW vector of “croutons” and the measure *Q* represents the BOW vector of “Mary” or “salad”. We like to see that the KL-divergence value of "croutons" and "salad" is smaller than the KL-divergence value of “croutons” and “Mary”. Therefore, we can determine that the prepositional phrase "with croutons" is more likely to be an attachment of "salad".

Notice that we did not use the information regarding the verb and the preposition in the sentence, which are usually useful information to disambiguate PPA in many cases. Our approach was focus on the study of BOW representation in PPA. Constructing a BOW vector is easily obtained, and avoids heavy work of feature engineering. We like to verify above assumption and demonstrate our approach to be a straightforward way to address the problem in PPA. Next we will discuss our experiment.

5.3 Results and Evaluation for Prepositional Phrase Attachment

We collected 30 sentences from PennTree Bank data set and previous papers, and manually extracted named entities in subject, object and prepositional phrase of each sentence. Then we used the method as described in Table 11. The proposed method correctly attached 22 out of 30 prepositional phrases and achieved 73.3% accuracy. Table 12 listed examples of results.

Table 12. Examples of PPA results

Sally watches the bees with her binoculars. The K-L divergence: <binoculars> vs <Sally> = 1.107 The K-L divergence: <binoculars> vs <bees> = 1.882
The doctor has approved the new treatment for his ailment. The K-L divergence: <aliment> vs <doctor> = 1.005 The K-L divergence: <aliment> vs <treatment> = 1.227
Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov 29. The K-L divergence: <director> vs <Pierre Vinken> = -0.138 The K-L divergence: <director> vs <board> = 2.721
The painting was presented to the audience by its author. The K-L divergence: <author> vs <painting> = 2.710 The K-L divergence: <author> vs <audience> = 2.459
Tom buy a car with a steering wheel. The K-L divergence: <wheel> vs <Tom> = 1.319 The K-L divergence: <wheel> vs <car> = 0.866
Mary ate the salad with a fork The K-L divergence: <fork> vs <Mary> = 2.376

The K-L divergence: <fork> vs <salad> = 2.278
Mary ate the salad with croutons. The K-L divergence: <croutons> vs <Mary> = 3.027 The K-L divergence: <croutons> vs <salad> = 2.069
Ben hang a painting with a nail. The K-L divergence: <nail> vs <Ben> = 0.869 The K-L divergence: <nail> vs <painting> = 1.718

In each item of the above table, it first showed a sentence, then listed KL-divergence values for

- <NE in PP, NE in subject>
- <NE in PP, NE in object>

Here NE means named entity, and PP means prepositional phrase.
We looked at the line with lower KL-divergence value and checked if PP attachment was correct.

To statistically evaluate our method, we defined a confidence score to be absolute difference between two KL-divergence values,

$$score = \left| KLD_2 - KLD_1 \right| \qquad (3)$$

KLD_1 and KLD_2 means the KL-divergence value of <NE in PP, NE in subject> and <NE in PP, NE in object> respectively. We computed the confidence scores for 30 sentences in our experiment, and then sorted the sentences by score values. Table 13 lists the sentences with confidence score in descending order. We checked correctness of the PP-attachment for each sentence. The sentences in Italic font mean our PPA method has made wrong decisions for them. We marked numbers of these sentences in "red" color.

Table 13. PPA results with sentences and their confidence scores.

No	Sentences	Confidence
1	Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov 29	2.859
2	Most of the stock selling pressure came from Wall Street professionals, including computer-guided program traders.	2.312
3	Improving profitability of U.S. operations is an extremely high priority in the company.	2.069
4	In July, the Environmental Protection Agency imposed a gradual ban on virtually all uses of asbestos.	1.122
5	Mary ate the salad with croutons.	0.958
6	Ben hangs a painting with a nail.	0.849

7	Sally watches the bees with her binoculars.	0.775
8	Sally watches the bee with the antennae.	0.706
9	I ate a pizza with friends.	0.696
10	Mr. Pope owns 10,000 UAL shares and has options to buy another 150,000 at $69 each.	0.625
11	The waitress prodding the clown with the umbrella.	0.613
12	He declined Sea Containers to raise its price.	0.542
13	The policeman prodding the doctor with the gun.	0.533
14	Steve saw Eric with a coat.	0.467
15	*Steve saw the stars with a telescope.*	*0.462*
16	In July, the Environmental Protection Agency imposed a gradual ban on virtually all uses of asbestos.	0.461
17	Tom buys a car with a steering wheel.	0.453
18	*Tom buys a car with his credit card.*	*0.443*
19	The doctor has approved the new treatment for his ailment.	0.359
20	*Ben hangs a painting with watercolor.*	*0.325*
21	*I ate a pizza with anchovies.*	*0.322*
22	Under an agreement signed by the Big Board and the Chicago Mercantile Exchange, trading was temporarily halted in Chicago	0.297
23	*The painting was presented to the audience by its author.*	*0.25*
24	I ate a pizza with pepperoni.	0.223
25	Tom buys a car with his wife.	0.206
26	He left his last two jobs at Republic Airlines and Flying Tiger with combined stock-option gains of about $22 million, and UAL gave him a $15 million bonus when it hired him.	0.174
27	Mr. Carlucci, 59 years old, served as defense secretary in the Reagan administration.	0.165
28	*You could say their business in the U.S. was mediocre, but great everywhere else.*	*0.159*
29	*Analysts estimate Colgate's sales of household products in the U.S. were flat for the quarter, and they estimated operating margins at only 1% to 3%.*	*0.1*
30	*Mary ate the salad with a fork.*	*0.099*

Looked through the errors, for instance, in the sentence “**Mary ate salad with a fork”,** we can think of the word distribution of “fork” is likely similar to the word distribution of “salad” than with “Mary”. The case for the sentence “**Steve saw the stars with a telescope”** is also the same**.** The BOW vector “telescope” is likely close to BOW vector “stars” than BOW vector “Steve". Other mistakes such as “**I ate a pizza with anchovies.”** and “**The painting was presented to the audience by its author.**”, “I” is very general and has many chances to appear with "anchovies” in the first sentence. On the other side, “anchovies” is not a very popular pizza top. It is not a surprise that the word distribution of “anchovies” is closer to “I” than to “pizza”. In another sentence, “its” in the prepositional phrase is important information to disambiguate “painting” and “audience”. However, it is hard to say that the word distribution of “painting” is close to the word distribution of “author” than “audience”. This is a limit of our proposed method.

Nevertheless, the top ranked sentences are correctly determined the pp-attachment with high confidence score, and incorrectly determined sentences are always have low confidence score, we can see that in general our proposed method gains a high precision for correctly determining the attachment of the prepositional phrase.

We still used precision (Eq. 1) in the top *N* of the ranking list as our evaluation metric. The precision of top 10, top 20 and top 30 are calculated and shown in Figure 8. The precisions in the top10, top20 and top30 according to the sorted confidence scores are 100%, 85% and 73% respectably.

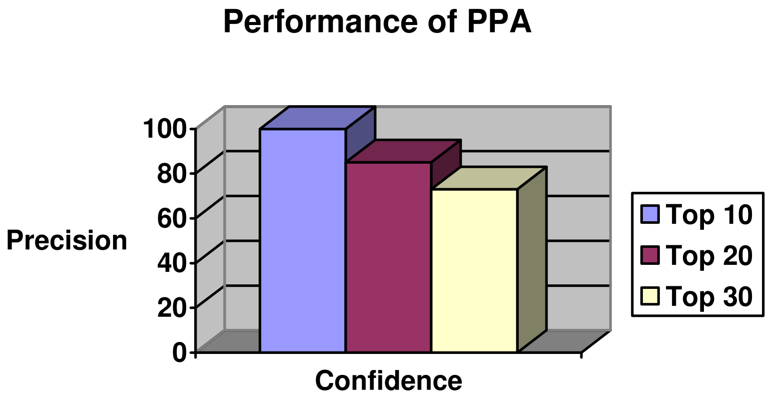


Figure 8. Performance of Prepositional Phrase Attachment

5.4 Summary of Prepositional Phrase Attachment

We have been discussed our implementation of Prepositional Phrase Attachment. Comparing with previous work, we proposed a novel and straightforward approach using our BOW representation. Our method is unsupervised, distance similarity measurement without any training. The results demonstrate that we can attain high precisions in 30 sentences of our experiment.

We also noticed that using limited grammar-based methods may have advantages in some cases. In a good future direction, we can try to model the entire phrase to see if it is helpful in improving the performance.

6 Conclusion

In this project, we have presented a preliminary study for semi-supervised discovery of named entities and relations within a read-the-web framework. We have proposed BOW representation to capture word distributions of any kind of NLP items on the web, such as named entities, relational tuples, phrases etc. Different methods to create a BOW vector have been illustrated and can be easily extended to characterize the semantic meaning of other language elements. Our BOW representation characterizes NLP items directly using the web by a search engine, which leverages the most representative and informative contextual features. The BOW vector is in uniform format and easy to use for machine learning algorithms.

The novelty of our BOW representation is that it interprets global probability distributions of any NLP items through a search engine. We only need sentences that contain a NLP item from a few numbers of top-ranked web pages to construct a BOW vector. Comparing with previous work of using BOW on a fixed corpus, our method is greatly efficient.

We have also investigated three research problems using our BOW representation, which including Named Entity Recognition, Relation Extraction and Prepositional Phrase Attachment. Different approaches have been particularly designed for three problems. The experiments show very high precisions on NER, RE and PPA by using our BOW technology.

Acknowledgments

I would like to thank my advisor Tom Mitchell. His earnest instructions and patience intrigued my passion of research. My every progress contains his motivation and encouragement. Also I greatly appreciate to Geoff Gordon, Steve Fienberg, Seyoung Kim and Kevin Killourhy for their helpful suggestions to improve the report. In particular, thanks for the support from Google to provide search API for our research. Finally, I am deeply grateful for my family's love and support.

References

ACE http://projects.ldc.upenn.edu/ace/data

Agichtein E. and Gravano L.. 2000. Snowball: Extracting Relations from Large Plain-Text Collections, *Proceedings of the Fifth ACM International Conference on Digital Libraries.*

Banko M., Cafarella M. J., Soderland S., Broadhead M., and Etzioni O.. 2007. Open Information Extraction from the Web, *Proceedings of IJCAI.*

Donald Hindle and Mats Rooth, 1991. Structural Ambiguity and Lexical Relations. *In Meeting of the Association for Computational Linguistics, pages 229–236.*

D. Nadeau. 2007. Semi-Supervised Named Entity Recognition: Learning to Recognize 100 Entity Types with Little Supervision. *PhD Thesis*.

E.Agichtein and L.Gravano. 2000 Snowball: Extracting relations from large plaintext collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries (DL-00)*, San Antonio, Texas.

E. Brill and P.Re snik. 1994. A transformation-based approach to prepositional phrase attachment disambiguation.In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, pages 1198–1204, Kyoto, Japan.

E. Riloff, R. Jones 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligenc.*

Google Search API, http://code.google.com/apis/ajaxsearch/web.html

Larry Wasserman, 2004, All of Statistics.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. *In Proceedings of the 14th International Conference on Computational Linguistics.*

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the 1999 Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*, pages 189–196, College Park, Maryland.

Michael Collins and James Brooks, Prepositional Phrase Attachment through a Backed-Off Model, *ACL 1995*

OpenNLP, http://opennlp.sourceforge.net/

Patrick Pantel, Dekang Lin An Unsupervised Approach to Prepositional Phrase attachment using Contextually Similar Words, *ACL 2000*.

R. Wang and W. Cohen: Language-Independent Set Expansion of Named Entities using the Web. In *Proceedings of IEEE International Conference on Data Mining* (ICDM 2007), Omaha, 2007.

Richard C. Wang and William W. Cohen: Iterative Set Expansion of Named Entities using the Web. In *Proceedings of IEEE International Conference on Data Mining* (ICDM 2008), Pisa, Italy. 2008.

R. Yangarber, W. Lin, and R. Grishman. 2002. Unsupervised learning of generalized names. In *Proc.COLING-2002*, Taipei, Taiwan.

Sanda M. Harabagiu. Patterns of Prepositional Attachments, 2000 - Where Dictionary Semantics Meets Corpus Statistics. In International Journal of Pattern Recognition and Artificial Intelligence, volume 14, pages 809–838.World Scientific Publishing Company.

S. Brin. 1998. Extracting patterns and relations from the World Wide Web. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT-98), Workshop on the Web and Databases*, pages 172–183, Valencia, Spain.

S. Cucerzan and D.Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence. In *Proceedings of the 1999 Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*, pages 90–99, College Park, Maryland.

T. Mitchell, S. Wang, Y. Huang, and A. Cheyer. 2006. Extracting knowledge about users activities from raw workstation contents. *In AAAI.*

T. Micthell, 2005. Reading the Web: A Breakthrough Goal for AI, *AI Magazine, Fall 2005, AAAI Press, Menlo Park, CA.*

Yarowsky D: Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics 1995

WIT (Web Intelligent Toolkit) http://www.cs.cmu.edu/~wit/WIT2.html