WebSets : Unsupervised Information Extraction approach to Extract Sets of Entities from the Web

[Extended Abstract] [*]

Bhavana Dalvi
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
bbd@cs.cmu.edu

William Cohen
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
wcohen@cs.cmu.edu

Jamie Callan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
callan@cs.cmu.edu

ABSTRACT

We propose an unsupervised information extraction system, which exploits the structured information in the form of HTML tables to build meaningful sets of entities belonging to certain categories. Due to redundancy on the Web, we believe that entities belonging to important categories will frequently co-occur in table columns. We present a clustering algorithm to cluster such frequently occurring entities into meaningful sets. Then we find candidate category names for each set, using Hearst patterns like "X such as Y", "X including Y".

Experimental results on four different datasets show that our method can extract meaningful sets of entities (with avg. cluster precision of 97-99%). It also proposes reasonable category names for them. We present an application of this method to enhance an existing knowledge base. Experiments show that our method improves the coverage of existing categories with 80-90% accuracy. It also suggests new categories that can be added to the knowledge base.

1. INTRODUCTION

Information extraction from structured as well as unstructured sources on the Web has been of interest in recent years. A new generation of information extraction systems like TextRunner[6] and NELL(Never Ending Language Learning)[4] is emerging that analyzes large corpora of unstructured text and creates structured information resources in the form of knowledge bases. Another variant of supervised learning algorithms like SEAL(Set Expansion for Any Language)[20] starts from small number of seeds and expands the set using a bootstrapping process. ASIA (Automatic Set Instance Acquirer)[19] starts with a concept name and finds out set of instances belonging to that concept.

Most of the existing systems need some human input to start the extraction process. Motivation behind our work is to devise a fully unsupervised information extraction technique. In this paper, we propose "WebSets", an unsupervised approach to the problem of extracting sets of entities from tables found on the Web. This approach builds meaningful sets of entities by exploiting structured information in the form of HTML tables that millions of web-page authors have created. We hypothesize that entities in same column of a table possibly belong to the same category. We verify this hypothesis by building an unsupervised system to do this IE task. Our system first extracts good quality tables from a set of web pages. It then finds the frequently co-occurring sets of entity triplets. Clustering these triplets we get sets of entities, which may belong to some meaningful category. The system also suggests category names for these clusters with the help of hyponym data built using open-domain Hearst patterns [11] like "X such as Y". We evaluate our method based on a comparison with existing extraction methods. We also evaluate the meaningfulness of generated clusters along with the accuracy of labels automatically assigned to the clusters.

One of the important applications of such set extraction techniques is to enhance an existing knowledge base. Additional experiments show that WebSets can contribute new instances of existing categories in a knowledge base and can also suggest new categories that can be added to make it richer.

The rest of this paper is organized as follows. Section 2 presents an overview of related work in the area. Section 3 describes the WebSets system proposed in this paper. Section 4 describes the evaluation methodology and experimental results of WebSets including its application to enhance an existing knowledge base. In section 5 we present our conclusions.

2. RELATED WORK

Information extraction from structured information sources on the Web is an area of active research in recent years. WebTables [2] was one of the first few systems, which proposed the use of tables on the Web as a source of high-quality relational data. It used the data from tables, to build the collection of attribute co-occurrence statistics, which in turn can be used to create a column thesaurus and column auto-completion system. Limaye et. al. [14] describes an

[*]A full version of this paper is available as *ACM SIG Proceedings Using $\LaTeX2_\epsilon$ and BibTeX* at `www.acm.org/eaddress.htm`

algorithm to annotate table cells and columns with corresponding categories and relations from an existing catalog or type hierarchy. Gupta and Sarawagi [10] address the problem of jointly training statistical extraction models for multiple Web sources to maximally exploit any content overlap. This technique tries to maximize the probability that the different sources agree on the labels of the overlapping content. Gupta et.al [9] focuses on the task of assembling a table given few example rows. Their technique consolidates HTML lists relevant to the example rows to build a result table. Using large datasets like AOL query logs, Parameswaran et. al. [15] proposes a concept extraction algorithm which can identify canonical form of concept and filters out any of its sub-concept or super-concept.

There has been a significant amount of work done in the area of web information extraction. Let us first consider the systems which take predicates as input to decide what kind of named entities to extract. KnowItAll [6, 7] is an unsupervised named-entity extraction system, which takes as input a set of domain specific predicates. The system then focuses on extracting those named entities which satisfy these predicates. Coupled Pattern Learning (CPL) [4] is a technique that enables the simultaneous semi-supervised training of category and relation extractors. The input to this system is an ontology that defines a set of target categories and relations to be learnt, few seed examples for each, and a set of relationships that couple the various categories and relations (e.g., Furniture and Mammal are mutually exclusive).

Another set of IE systems do not need predicates or ontologies as input. SEAL [20] demonstrated a set expansion technique, which starts with set of seed examples for each set and expands them by finding remaining instances of the set. This system builds "wrappers" for the semi-structured webpages using existing instances, uses those wrappers to find more instances, and continues this iteratively to expand the set of entities. ASIA [19] further reduces the human input requirement by just taking the category names as input and then extracting sets of named entities belonging to those categories using SEAL as one of its components.

Coupled SEAL (CSEAL) [4] makes use of mutual exclusion, containment and type checking relationships on top of conventional SEAL. Furthermore, CPL and CSEAL are the two most important components of the NELL system [18]. The term "NELL knowledge base" used hereafter in this paper, refers to the KB which consists of (1) manually created set of categories and relations (2) seed examples for each category/relation (3) relationships between these categories/-relations and (4) additional instances of these categories/-relations that are learnt by CPL and CSEAL.

TextRunner [21] is an Open Information Extraction system in which the system makes a single, data-driven pass over the entire corpus and extracts a large set of relational tuples, without requiring any human input. It also embeds synonym resolution mechanism, to collapse multiple noun-phrases which refer to the same entity. However TextRunner does not build consistent sets of category or relations.

Gatterbauer et. al. [8] discusses approaches to extract various forms of tables which might not have tree like structure of HTML tables but when the page is rendered in a browser, it looks like table. Their work focuses on extracting tabular data from various kinds of visual representations. Given an arbitrary noun-phrase, Ritter et. al.[16] describes a technique which first generates candidate hypernyms using co-occurrence statistics on a large corpus. It then runs an SVM classifier on these candidates using Hearst pattern occurrences and POS tag features to decide which candidate hypernyms are correct.

Our approach is different from most of the earlier approaches in the sense that we start with structured data like HTML tables and build consistent sets of entities, without any human input. We then use Hearst patterns and co-occurrence based statistics, to propose candidate category names for the set.

3. PROPOSED APPROACH

Tables on the Web often contain lists that define meaningful sets. We feel that an Information Extraction system can make a wise use of this already available structured information to come up with sets of entities hidden in the webpages. The sets extracted can also be seen as a summary telling what concepts are talked about in those webpages. In this paper, we propose an unsupervised information extraction technique WebSets to address this problem.

WebSets is composed of four main components: the *Table Parser*, the *Triplet Store Builder*, the *Bottom-Up Clusterer*, the *Hypernym Recommender*. Given a set of HTML pages, the *Table Parser* extracts potentially useful tables using a set of simple hand-coded heuristics. The *Triplet Store Builder* then goes over every column of every extracted table, and builds a dataset of entity triplets found in them. Each record of triplet store contains entities in the triplet, all TableID:ColumnId's in which they co-occurred, and all domains in which they co-occurred. These triplets are then ranked by number of distinct domains in which they co-occurred. The *Bottom-Up Clusterer* then clusters these triplets into consistent sets. This component considers only those triplets that have occurred in at least k number of domains, where k is a small constant. The *Hypernym Recommender* then considers each cluster, and recommends candidate hypernyms based on the overlap between the cluster and hypernym pattern dataset.

Each component of this unsupervised pipeline, enhances the quality and completeness of resultant sets of entities. In this section, we describe each component in detail.

3.1 Table Parser

While doing data analysis, we found that most HTML pages use tables for formatting or rendering purposes and hence do not necessarily contain relational data. Another issue we faced was that lot of HTML pages have broken HTML syntax, which creates problems while parsing them correctly.

We used an HTML syntax cleaning software Tidy [1] which fixes the HTML syntax in these pages. *Table Parser* is then run on these clean HTML pages. To filter out tables with useful relational data, *Table Parser* uses following set of features from each table : (1) The table is recursive (if yes discard it) (2) The number of rows and columns (at least 3 rows and 2 columns) (3) The number of non-link columns (at least 2 non-link columns) (4) The length of cells after removing formatting tags (allowed length is 2 to 50 char).

We manually set thresholds for these features, to separate out reasonable quality tables. These thresholds remain constant for all the experiments described in this paper. Currently *Table Parser* parses only HTML tables which have $< table >$ tags in them. This is only a fraction of struc-

Country	Capital City
India	Delhi
China	Beijing
Canada	Ottawa
France	Paris

Table 1: Example Table (TableId = 10 , URL = "http://www.dummy.com/index.html")

Entities	Tid:Cids	Domains
India,China,Canada	10:0	www.dummy.com::1
China, Canada, France	10:0	www.dummy.com::1
Delhi, Beijing, Ottawa	10:1	www.dummy.com::1
Beijing, Ottawa, Paris	10:1	www.dummy.com::1

Table 2: Triplet records created by WebSets

tured data available on the Web. Use of other techniques like Gatterbauer et. al. [8] can provide more input data to learn sets from.

3.2 Triplet Store Builder

After filtering out potentially useful HTML tables, the system needs to find out which entities in these tables belong to some meaningful category. To solve this problem we propose a method based on co-occurrence of entity triplets. The choice of working on triplets instead of individual entities is based on an independent observation by Wang et. al.[20]. During evaluation of SEAL, it was found that, given a set of three seed entities belonging to a category, the system can find remaining entities with 90% MAP. This shows that a set of three entities can give reasonable information about the category they belong to.

Triplet Store Builder goes over every column of every extracted table, and builds a dataset of entity triplets found in them. Each record of triplet store contains entities in the triplet, all TableID:ColumnId's in which they co-occurred, and all domains in which they co-occurred. For a table column containing n entities, considering all possible triplets will result in $O(n^3)$ triplets. This will considerably increase the size of the triplet store. Hence we only consider triplets which are size three sub-sequences of sequence of entities in each table column.

Consider an example table containing countries and their capitals. Original table and the triplet records created by the *Triplet Store Builder* are shown in Table 1 and Table 2. Second row in Table 2, indicates that the triplet (China, Canada, France) occurred in TableId = 10, ColumnId = 1, on a webpage which resides in the domain "www.dummy.com". Similarly, "www.dummy.com::1" denotes that the triplet occurred in this domain only once. If a triplet occurs in multiple tables or domains, all of them will be stored in the triplet record.

When we actually store these triplets, we canonicalize them by converting them to lower case and arranging the constituent entities in alphabetical order. The system keeps track of all TableId:ColumnId a triplet occurred in, which makes it possible to reconstruct a column by joining triplets on TableId:ColumnId. Hence this storage method does not result in any loss of information.

After scanning all tables from all HTML files, a huge list of entity triplet records is produced. The *Triplet Store Builder* ranks these triplets according to number of domains they were found in. We create $O(n)$ triplets from a table column of size n. Adding each triplet to the Triplet Store using hashmap takes $O(1)$ time. Given a set of T html tables with a total of N entities in them, the Triplet Store can be created in $O(N)$ time. Ranking the triplets using any common sorting technique will take $O(N * logN)$ time. Hence total complexity of the *Triplet Store Builder* is $O(N * logN)$

3.3 Bottom-Up Clusterer

Bottom-Up Clusterer scans all records of the triplet store and clusters them into consistent sets of entities. This is an unsupervised system and we do not know in advance how many clusters are present in the triplet store. Hence we can not use parametric clustering algorithm as k-means clustering. Non-parametric algorithms like agglomerative clustering [5] fits our requirements. The *Bottom-Up Clusterer* uses a variant of agglomerative clustering for the task of clustering entity triplets.

3.3.1 Algorithm

Refer to Algorithm 1 for a formal description of the algorithm. The clusterer scans through each triplet record t which has occurred in at least k distinct domains. This makes sure that it considers only those triplets which are considered meaningful across multiple domains. A triplet and a cluster are represented with the same data-structure, which consists of set of entities, a set of TableId:ColumnId in which entities co-occurred and a set of domains that entities co-occurred in.

Clusterer compares the overlap of triplet t against each cluster C_i. Triplet t is merged in C_i if either of the following two cases is true :
(1) If any two entities from t appear in already existing cluster C_i
(2) If any two TableId:ColumnId from t appear in already existing cluster C_i.
In both these cases, intuitively there is a high probability that t belongs to same category as C_i.
If no such overlap is found with existing clusters, algorithm creates a new cluster and initializes it with the triplet t.

3.3.2 Computational complexity

Suppose that our dataset has total T tables and N entities in the cells of those tables. For each triplet t, Algorithm 1 finds entity and TableId:ColumnId overlap with all existing clusters. This operation can be implemented efficiently by keeping two inverted indices : (1) from an entity to all clutserIds it belongs to and (2) from each TableId:ColumnId to all clusterIds it belongs to. ClusterIds in each posting list will be kept in sorted order.

Merging k sorted lists, with resultant list size of n takes $O(n * logk)$ time. Now let us compute worst case time complexity of Algorithm 1. To compute entity overlap of each triplet, we will merge 3 posting lists with total size of $O(N)$. This step will take $O(N)$ time. To compute TableId:ColumnId overlap of each triplet, we will merge $O(T)$ posting lists with total size of $O(N)$. This step will take $O(N * logT)$ time. Hence for each triplet, finding the clusterId to merge a triplet takes $O(N * logT)$ time. There are $O(N)$ triplets. So the *Bottom-Up Clusterer* will have worst case time complexity of $O(N^2 * logT)$.

Algorithm 1 Bottom-Up Clustering Algorithm

```
 1: function Bottom-Up Clusterer ($TripletStore$) : $Clusters$
 2: Initialize $Clusters = \phi$; $max = 0$
 3: for (every $t \in TripletStore$ : $|t.domains| >= k$) do
 4:   assigned = false
 5:   for every $C_i \in Clusters$ do
 6:     if $|t.entities \cap C_i.entities| >= 2$ OR
        $|t.TidCid \cap C_i.TidCid| >= 2$ then
 7:       $C_i = C_i \cup t$
 8:       $assigned = true$; break;
 9:     end if
10:   end for
11:   if not assigned then
12:     increment $max$
13:     Create new cluster $C_{max} = t$
14:     $Clusters = Clusters \cup C_{max}$;
15:   end if
16: end for
17: end function
```

Hyponym	Concepts
USA	country:1000
India	country:200
Paris	city:100, tourist_place:50
Monkey	animal:100, mammal:60
Sparrow	bird:33

Table 3: Example records in Hyponym Concept Dataset

3.4 Hypernym Recommender

This component of the system recommends candidate hypernyms for each entity set produced by the *Bottom-Up Clusterer*. For this task, we created a *Hyponym Concept Dataset*, which is derived from Clueweb dataset. Detailed description of how the *Hyponym Concept Dataset* was derived is given in Appendix A.

Each record in this dataset contains an entity and all concepts it co-occurred along with co-occurrence counts. Table 3 shows a dummy example of records in this dataset. According to this table, entity “USA” appeared with the concept “country” 1000 times. Similarly, “Monkey” appeared with two different concepts, 100 times with “animal” and 60 times with “mammal”.

For each set produced at the end of clustering, the *Hypernym Recommender* finds which entities from the set belong to *Hyponym Concept Dataset*, collects all concepts they cooccur with. Then these concepts are ranked by number of unique entities in the set it co-occurred with. This ranked list serves as hypernym recommendations for the set. Note that this part of the system uses information extracted from unstructured text from the Web, to recommend category names to sets extracted from tables on the Web.

The *Hyponym Concept Dataset* described above can be noisy ,i.e., it may contain hyponyms which are not meaningful entities. This noise can get introduced due to various issues including but not limited to, sentence parsing and segmenting errors, spam on the Web etc. A Knowledge Base (KB) can be used to restrict hyponyms to only the valid entities in KN. This will generate a high quality Hyponym Concept Dataset but it will have very low coverage.

Dataset	#HTML pages	#Useful tables	#triplets
DS-1	30K	116K	1148K
DS-2	112K	233K	421K
DS-3	121K	216K	374K
DS-4	100K	176K	78K

Table 4: Dataset Statistics

4. EXPERIMENTAL EVALUATION

While evaluating performance of WebSets, we want to answer three main questions:
(1) What fraction of the generated clusters are meaningful?
(2) How precise are the meaningful clusters? I.e. what fraction of the entities in a cluster belong to the category name assigned to that cluster.
(3) Does the *Hypernym Recommender* generate reasonable category names?

In order to answer these questions, we run WebSets in unsupervised setting and do manual evaluation of meaningfulness and precision of clusters.

4.1 Datasets

To test the performance of WebSets, we created focused datasets of webpages which are likely to have consistent sets of entities. An evaluation can then be done to check whether the system extracts expected sets out of those datasets.

1. **DS-1 (SEAL_Useful) :** This dataset is a collection of 30K pages in which SEAL found useful entities which are part of NELL KB. SEAL mostly extracts entities out of semi-structured information on the Web, which include HTML tables. Hence we expect that WebSets will extract good quality sets out of this dataset.

2. **DS-2 (ASIA_NELL) :** This dataset is collected using the ASIA system. Frequent hypernyms associated with entities in the NELL KB were used as queries on ASIA. Examples of hypernyms in this domain are “City”, “Bird”, “Sports team” etc. ASIA [19] is a version of SEAL which takes a category name as input and finds possible instances of that category using set expansion techniques. It stores all the pages downloaded during this process in a cache, so that they can be reused for any similar queries in the future. The cache contents generated by a set of related ASIA queries can serve as reasonable dataset for our experiments.

3. **DS-3 (ASIA_INT) :** This dataset is collected using the ASIA system by giving another set of category names as input. These category names come from “Intelligence domain”. Examples of categories in this domain are “government types”, “international organizations”, “federal agencies”, “religions” etc.

4. **DS-4 (Clueweb_HPR) :** This dataset is collected by randomly sampling pages in the Clueweb dataset [3] with spamrank score higher than 60%. For this purpose we used the Fusion spam scores provided by Waterloo university [12].

Table 4 shows the number of tables and triplets extracted from each of these datasets. Recall from Section 3 that during processing, HTML pages in all these datasets are cleaned using Tidy software [1]. Also note the difference between DS-4 and remaining datasets. DS-4 contains random sample of the Clueweb dataset, and its considerable fraction may not contain good relational tables. On the other hand, remaining datasets are derived using either SEAL or ASIA, which work on structured text like lists, tables etc. This might explain comparatively lower number of useful tables and triplets extracted from DS-4.

4.2 Mechanical Turk Evaluation

Mechanical Turk has been shown to be an inexpensive and fast method for obtaining labels for language tasks [17]. Subjective evaluation like "deciding whether a cluster is meaningful or noisy", "assigning label to an unlabeled cluster" is done with the help of our colleagues, we refer to them as evaluators.

Objective evaluation of the kind "whether X belongs to category Y" is done using Mechanical Turk. We created yes/no questions of the form "Is X of type Y?". To evaluate precision of clusters created by WebSets, we uniformly sampled maximum 100 clusters per dataset, with maximum of 100 samples per cluster and gave them to the Mechanical Turk in the form of yes/no questions. Each question was answered by three different individuals. The majority vote for each question was considered as a decision for that question. We evaluated quality of labels produced by Mechanical Turk, to make sure that precision estimates we present are not biased high. We observed that labels are accurate more than 90% times and precision estimates are biased low. For more details about this evaluation refer to Appendix B.2.

4.3 Experimental Results

In this section we will see the results of evaluating WebSets using three criteria : (1) Are the clusters generated by WebSets meaningful? (2) What is precision of the meaningful clusters? (3) How good are recommended hypernyms? Subsequent sections will discuss the experiments in detail.

4.3.1 Meaningfulness of Clusters

In this experiment, we did manual evaluation of meaningfulness of clusters, with the help of evaluators. We uniformly sampled maximum 100 clusters from each dataset. We showed following details of each cluster to the evaluator : (1) top 5 hypernyms suggested by the *Hypernym Recommender* (2) maximum 100 entities sampled uniformly from the cluster.

An evaluator was asked to look at the entities and check whether any of the hypernyms suggested by the system is correct. If any one of them is correct then he labels the cluster with that hypernym. If none of the hypernym is correct, he can label cluster with any other hypernym that represents the cluster. If entities in a cluster are noisy or do not form any meaningful set, then the cluster is marked as "noisy". If the evaluator picks any of the candidate hypernyms as label or gives his own label then the cluster is considered as meaningful, else it is considered as noisy.

Table 5 shows that 63-73% of the clusters were labeled as meaningful. Note that number of triplets used by the clustering algorithm (Table 5) is different from total number of triplets extracted by the *Triplet Store Builder* (Table 4). This is due to the fact that the *Bottom-Up Clusterer* considers only those triplets which occurred in at least k distinct domains. We set $k = 2$ for these experiments.

	#Triplets	#Clusters	#clusters with Hypernyms	% meaningful
DS-1	165.2K	1090	312	69%
DS-2	11.4K	448	266	73%
DS-3	15.1K	395	218	63%
DS-4	561	47	34	70.5%

Table 5: Meaningfulness of generated clusters

Dataset	#Meaningful clusters evaluated	% Precision
DS-1	69	98.6%
DS-2	73	98.5%
DS-3	63	97.4%
DS-4	24	99%

Table 6: Avg. precision (%) of meaningful clusters

4.3.2 Precision of meaningful clusters

In this experiment we want to evaluate following two things:
(1) How consistent are the clusters? I.e. do all entities in a cluster belong to the category specified by manual label assigned to the cluster?
(2) How accurate is the manual labeling of clusters?

To answer these two questions, we evaluated the meaningful clusters found in previous experiment, using the Mechanical Turk. This evaluation procedure is already discussed in Section 4.2. Table 6 shows that the meaningful clusters have precision in the range 97-99%. This indicates that those clusters are indeed consistent and and cluster labels of meaningful clusters assigned by the evaluators are of good quality.

4.3.3 Performance of Hypernym Recommender

In this experiment, we evaluate the performance of the *Hypernym Recommender* using following criterion:
(1) What fraction of total clusters were assigned some hypernym? : This can be directly computed by looking at the outputs generated by the *Hypernym Recommender.*
(2) For what fraction of clusters evaluator chose the label from the recommended hypernyms? : This can be computed by checking whether each of the manually assigned labels was one of the recommended labels.
(3) What is Mean Reciprocal Rank (MRR) of the hypernym ranking? : The evaluator gets to see ranked list of top 5 labels suggested by the *Hypernym Recommender.* We compute MRR based on rank of the label selected by the evaluator. While calculating MRR, we consider all meaningful clusters(including the ones for which label does not come from the recommended hypernyms).

Table 7 shows the results of this evaluation. Out of the random sample of clusters evaluated, *Hypernym Recommender* could label 50-60% of them correctly. MRR of labels is 0.56-0.59 for all the datasets. There is definitely a chance of improvement in this component of the system. We use very simple technique based on co-occurrence of entities and concepts. However our system is modular and in future we plan to plug in more complex techniques like Ritter et. al. [16] to improve the performance of the *Hypernym Recommender.*

	#Clusters Evaluated	#Meaningful Clusters	#Hypernyms correct	MRR (meaningful)
DS-1	100	69	57	0.56
DS-2	100	73	66	0.59
DS-3	100	63	50	0.58
DS-4	34	24	20	0.56

Table 7: Evaluation of Hypernym Recommender

4.4 Application : Adding to an existing Knowledge Base

One of the applications of WebSets lies in enhancing any existing knowledge base. It can contribute new instances of the existing categories and can also suggest new categories that can be added to make it richer. We used NELL knowledge base [18] for our experiments. As described in Section 2, the NELL knowledge base is created using the Clueweb corpus and contains around 518K beliefs. NELL has an underlying ontology which contains description of a set of categories, relations. It also includes few seed examples of each category and relation.

4.4.1 Semi-supervised Bottom-Up Clusterer

We changed the *Bottom-Up Clusterer* slightly for the task of enhancing a knowledge base. The Clusterer initializes *Clusters* with existing categories in the knowledge base. For experiments in this paper, we use around a dozen seed examples in each category of NELL for initialization. With this simple modification *Bottom-Up Clusterer* starts from NELL KB categories and end up with clusters which contain additions made to already existing KB clusters and suggestions of some new clusters. New entities added to existing NELL KB clusters, are referred by the term "promotions". This version of the algorithm uses facts already learnt by a knowledge base, hence categorized as Semi-supervised.

4.4.2 Experimental Evaluation using NELL

The set of experiments explained in this section, will enable us to answer following questions: (1) How precise are the recommendations made for existing categories of a knowledge base? (2) What is the coverage of WebSets system? Does it find the expected sets of entities? (3) How meaningful and precise are the new categories suggested for a knowledge base?

To answer these questions, we ran the WebSets system in semi-supervised mode. Coverage is measured by looking at promotions made by WebSets and other IE techniques (CPL and CSEAL) for same set of NELL categories. Manual evaluation process for measuring precision and meaningfulness of clusters is same as discussed in Section 4.3.

CPL extracts information out of sentences i.e. unstructured text, whereas WebSets use structured data like tables to extract entity sets. Both CSEAL [4] and WebSets use structured information to come up with entity sets, but the process they follow have some fundamental differences. CSEAL starts with seeds for each category and queries the Web for pages containing co-occurrence of seed instances. It then extracts more instances of same category from those pages by building wrappers, and the process continues. On the other hand, WebSets starts with set of all available HTML pages and comes up with sets that the system strongly believes are meaningful. These differences in approaches and nature of inputs indicate that direct comparison between WebSets and other IE techniques is not possible. Hence we compare WebSets and other IE systems in terms of their usefulness in enhancing existing knowledge base.

Category	#Promotions by WebSets			
	DS-1	DS-2	DS-3	DS-4
academicfield	-	3	9	-
athlete	2	-	-	-
boardgame	-	18	-	-
city	3117	90	245	-
coach	11	-	-	-
company	336	-	-	-
country	1569	262	778	19
emotion	-	4	-	-
hobby	7	13	-	-
mammal	-	8	-	-
politician	-	-	26	-
scientist	423	-	-	-
sportsteam	88	-	-	-
stateorprovince	106	41	38	31
university	-	-	9	-
Total	5659	439	1105	50

Table 8: Number of promotions by WebSets to existing NELL categories

Category	#Promotions	
	CPL	CSEAL
academicfield	46	203
athlete	132	276
boardgame	10	126
city	1000	368
coach	188	619
company	1000	245
country	1000	130
emotion	483	183
hobby	357	77
mammal	224	154
politician	178	30
scientist	83	928
sportsteam	301	864
stateorprovince	202	114
university	1000	961
Total	6204	5278

Table 9: Number of promotions by CPL and CSEAL to existing NELL categories

4.4.3 Coverage of Promotions

In this experiment, we compare the coverage of sets produced by WebSets with other IE systems CPL and CSEAL. We did this evaluation for only those NELL categories for which results of CPL and CSEAL are available in Carlson et. al. [4]. Number of promotions made by WebSets using different datasets are shown in Table 8. It shows categories for which WebSets found nonzero promotions using at least one of the four datasets. A missing entry in the table indicate WebSets does not produce any promotions for that particular category and dataset. Table 9 shows the number of promotions for CPL and CSEAL methods. They are

Category	Precision (%) of WebSets			
	DS-1	DS-2	DS-3	DS-4
academicfield	-	100	88.9	-
athlete	100	-	-	-
boardgame	-	72.2	-	-
city	84.5	82.2	85.5	-
coach	100	-	-	-
company	97	-	-	-
country	60.2	82.4	67.5	84.2
emotion	-	75	-	-
hobby	71.4	84.6	-	-
mammal	-	50	-	-
politician	-	-	92.3	-
scientist	88	-	-	-
sportsteam	97.7	-	-	-
stateorprovince	53.9	95.1	94.7	90.3
university	-	-	100	-
Avg.	83.6	80.2	88.1	87.2

Table 10: Precision of promotions by WebSets to existing NELL categories

Category	Precision(%)	
	CPL	CSEAL
academicfield	70	90
athlete	87	100
boardgame	80	70
city	97	97
coach	93	100
company	97	100
country	57	97
emotion	77	87
hobby	77	77
mammal	83	93
politician	80	97
scientist	97	100
sportsteam	90	87
stateorprovince	77	83
university	93	100
Avg.	83.6	91.8

Table 11: Precision of promotions by CPL and CSEAL to existing NELL categories

put in a separate table to indicate that the numbers are not directly comparable since they run on different corpuses.

DS-1 is the subset of data used by CSEAL and hence it behaves very similar to CSEAL in terms of total number of promotions. DS-2 leads to smaller number of promotions when compared to DS-1. This can explained by the fact that DS-1 consists of those pages from which CSEAL found NELL KB entities and hence are of higher quality than the pages in the cache of ASIA system. DS-3 largely contains information from intelligence domain, and does not cover many pages needed to get overlap with NELL clusters. Very few promotions from DS-4 can be attributed to the fact that it is a random sample from Clueweb, hence likely to have pages from diverse topics and not much redundancy for any specific topic.

We can also see that number of entities promoted by WebSets on different datasets, has a large variance. This is because each dataset has a different focus. Coverage of WebSets is lower than CPL and CSEAL when compared at the level of each NELL category. Please note that CPL runs on Clueweb [3] dataset (500M webpages), CSEAL issues queries on the Web, while WebSets is running on datasets of size 30-100K webpages. Taking these corpus differences into account, the performance of WebSets is encouraging.

4.4.4 Precision of Promotions

To compute precision of WebSets, we gave 100 random samples from each cluster found in each dataset to the Mechanical Turk and calculated precision of whole cluster based on precision of the sample. Similar to previous experiment, Table 10 and Table 11 show the precision evaluation for WebSets and other IE methods (CPL, CSEAL) respectively. From these results we can say that WebSets produces results with precision close to CPL and slightly less precision than CSEAL. More precision results of WebSets on remaining NELL categories are presented in Appendix B.1.

4.4.5 Evaluation of new category suggestions

In this section, we will evaluate both meaningfulness and precision of the additional clusters proposed by WebSets.

	#Clusters	#Meaningful	#Noisy	Avg. Precision (meaningful)
DS-1	107	81	26	94.44
DS-3	126	50	76	92.41

Table 12: Evaluation of meaningfulness of extra clusters suggested to NELL

These are the clusters which could not get merged with NELL clusters due to lack of entity overlap or absence of the category in NELL KB. To judge meaningfulness of these clusters we asked an evaluator to manually label each cluster after looking at the entities in it. If the cluster does not represent any consistent set, it is marked as "noisy".

Table 12 shows these evaluation results for datasets DS-1 and DS-3. We can see that 81 out of 107 clusters from DS-1 are meaningful with average precision within those clusters being 94.44%. However for DS-3, relatively smaller fraction of clusters (50 out of 126) are meaningful, but within meaningful clusters precision is quite high i.e. 92.41%. Poor performance on DS-3 suggests that the system needs some mechanism to rank meaningful clusters above noisy clusters. Features like "#entity matched in hyponym pattern dataset", and "#domains the triplets come from" can be used for this purpose.

Table 13 shows a sample of meaningful clusters from DS-1 and DS-3. We can see that most of the clusters have very high accuracy (above 95%) while few clusters have very low accuracy (bellow 50%). These results show that some categories are very much prone to noise. This suggests a need to strengthen our algorithm to avoid bridging between very different sets of entities that have a reasonable overlap. Table 13 also shows the diversity of cluster labels across DS-1 and DS-3 (note the highlighted labels). DS-1 is focused on pages containing NELL entities hence generates common categories like like "city", "bird", "sports team". However DS-3 is focused on intelligence domain, and could generate some very specialized categories like "religions", "international organizations", "social metrics", "police designa-

DS-1		DS-3	
Manual label	Precision	Manual label	Precision
tv channels	100	**airline**	100
tourist place	100	apparel	100
shoes/footwear	100	car make	100
researcher	100	**n/w protocol**	100
months	100	**governmentType**	100
laptop	100	iraq event	100
language	100	occupation type	100
pond	100	**police desgn**	100
celebrity	100	region	100
bird	100	**religion**	100
baseball players	100	contact info.	100
athletics	100	university	100
actress/models	100	**Intl.organization**	96.67
pond/lake	97.78	service	96.15
chemicals	97.78	department	93.33
product	96	linux jargon	92.86
mountain	91.67	province	88.24
genes	91.11	chemical element	87.38
book type	87.5	action	63.04
shoes	85.71	**social metric**	55.56
film studio	78.57	USA state	48.42
hobby	55.56	computer h/w	16.67

Table 13: Precision (%) of manually labeled extra clusters from DS-1, DS-3

tions" etc. These experiments with WebSets for enhancing the NELL KB do show encouraging results.

5. CONCLUSION

In this paper, we presented an unsupervised information extraction technique called WebSets, which exploits structured data in the from HTML tables on the Web. Our experiments with different datasets show that WebSets can extract meaningful sets with very high precision (97-99%). It also suggests reasonable category names for these sets. Further, WebSets can be used for the task of enhancing an existing knowledge base. Experiments with the NELL KB show that WebSets can suggest new entities for existing categories with reasonably high precision (80-87%). It also has the ability to find new clusters that can be added to NELL. Existing IE techniques like CPL and CSEAL do not have this capability.

In this paper we worked on extracting sets belonging to particular categories. As an immediate next step, we want to extend this system for extracting relations from tables on the Web. We also plan to integrate the semi-supervised version of WebSets with the NELL knowledge base. For this purpose, we will run WebSets on ClueWeb dataset to extract candidate instances and new category suggestions for NELL.

6. REFERENCES

[1] Html tidy library project. http://tidy.sourceforge.net/.

[2] M. J. Cafarella, E. Wu, A. Halevy, Y. Zhang, and D. Z. Wang. Webtables: Exploring the power of tables on the web. *PVLDB*, 2008.

[3] J. Callan. The clueweb09 dataset. http://boston.lti.cs.cmu.edu/Data/clueweb09/.

[4] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka, Jr., and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM*, 2010.

[5] W. H. E. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. In *Journal of Classification*, 1984.

[6] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in knowitall: (preliminary results). In *WWW*, 2004.

[7] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yate. Unsupervised named-entity extraction from the web: An experimental study. In *AI*, 2005.

[8] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. Towards domain-independent information extraction from web tables. In *WWW*, 2007.

[9] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. In *VLDB*, 2009.

[10] R. Gupta and S. Sarawagi. Joint training for open-domain extraction on the web: exploiting overlap when supervision is limited. In *WSDM*, 2011.

[11] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *ACL*, 1992.

[12] J. Kamps, R. Kaptein, and M. Koolen. Using anchor text, spam filtering and wikipedia for web search and entity ranking. *TREC*, 2010.

[13] Z. Kozareva and E. Hovy. A semi-supervised method to learn and construct taxonomies using the web. In *EMNLP*, 2010.

[14] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *PVLDB*.

[15] A. Parameswaran, H. Garcia-Molina, and A. Rajaraman. Towards the web of concepts: Extracting concepts from large datasets. In *VLDB*, 2010.

[16] A. Ritter, S. Soderland, and O. Etzioni. What is this, anyway: Automatic hypernym discovery. In *AAAI*, 2009.

[17] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *EMNLP*, 2008.

[18] M. Tom. Nell: Never-ending language learning. http://rtw.ml.cmu.edu/rtw/.

[19] R. C. Wang and W. W. Cohen. Automatic set instance extraction using the web. In *ACL*, 2009.

[20] R. C. Wang and W. W. Cohen. Character-level analysis of semi-structured documents for set expansion. In *EMNLP*, 2009.

[21] A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead, and S. Soderland. Textrunner: Open information extraction on the web. In *NAACL HLT*, 2007.

Id	Regular expression
1	arg1 such as (w+ (and|or))? arg2
2	arg1 (w+)? (and|or) other arg2
3	arg1 include (w+ (and|or))? arg2
4	arg1 including (w+ (and|or))? arg2

Table 14: Regular expressions used to create Hyponym Concept Dataset

APPENDIX

A. HYPONYM CONCEPT DATASET

To build the *Hyponym Concept Dataset*, we used data extracted from the ClueWeb09 corpus by the developers of the NELL system [18]. They used heuristics to identify and then shallow-parse approximately 2 billion sentences, and then extracted from this all patterns of the form "_ $word_1$ .. $word_k$ _" where the 'filler' $word_1$ .. $word_k$ is between one and five tokens long, and this filler appears at least once between two base noun phrases in the corpus. Each filler is paired with all pairs of noun phrases that bracket it, together with the count of the total number of times this sequence occurred. For instance, the filler '_ and vacations in _' occurs with the pair 'Holidays, Thailand' with a count of one and 'Hotels,Italy' with a count of six, indicating that the phrase "Hotels and vacations in Italy" occurred six times in the corpus . The hyponym dataset was constructed by finding all fillers that match one of the regular expressions in Table 14. These correspond to a subset of the Hearst patterns used in ASIA [19] together with some "doubly anchored" versions of these patterns [13].

B. MORE EXPERIMENTS

B.1 Performance of WebSets on the remaining NELL categories

In Section 4.4.4, we presented precision results for only those categories for which we already have precision values for CPL and CSEAL. In this section, we measure precision of promotions made by Websets to remaining NELL categories. Here, we evaluate the promotions against NELL category names. We sample 100 instances at random from each cluster from each dataset, and do the Mechanical Turk evaluation.

Results for all four datasets are shown in Table 15. As can be seen, for most of the clusters, precision is very high. WebSets gives noisy results few two clusters including "color" and "female names" when run on DS-3. We did some error analysis to understand, why the words like "they", "said", "no", "people", "are", get added to category "color". We found that several educational websites for children contain tables with all common words, which include the colors like "green", "red", "blue". As there are several such webpages (e.g. http://www.mrsperkins.com/dolch-words-all.html), WebSets promote several common words as "colors". This suggests the need for strengthening our clustering algorithm to avoid such cases.

B.2 Evaluating Quality of Mechanical Turk Experiments

To check the quality of the Mechanical Turk evaluation, we sampled 100 questions at random, and manually answered the questions. Then we checked whether majority vote by the Mechanical Turk matches with our answers. We specifically checked majority votes for some confusing questions which were likely to get labeled wrong. We found that majority vote of three individuals was correct more than 90% times. The only mistakes were false negatives, i.e. "X is of type Y", but popular vote said "no". This was mainly because X was not much popular(some very old scientist's name) or disputed(it was difficult to tell whether an island is country by itself or is part of some other ruling country). Except such cases, majority vote of three individuals was always correct. Since the mistakes are mainly false negatives, precision estimates we derive would not be biased high.

Category	Precision (%)			
	DS-1	DS-2	DS-3	DS-4
automobilemaker	-	100.00	83.33	-
automobilemodel	-	-	100.00	-
bird	80.00	100.00	-	-
blog	98.80	-	-	-
buildingmaterial	-	91.30	84.62	-
cardgame	-	100.00	-	-
chemical	88.65	94.87	-	-
color	-	71.43	**29.35**	-
consumerelectronicitem	-	73.68	-	-
continent	82.35	100.00	76.92	-
dayofweek	-	-	-	100.00
ethnicgroup	97.78	-	-	-
female names	-	-	**20.00**	-
fish	94.85	94.44	-	-
insect	100.00	-	-	-
language	88.29	100.00	92.93	100.00
magazine	-	-	100.00	-
mountain	100.00	-	-	-
musicgenre	100.00	-	-	-
musicinstrument	-	100.00	-	-
newspaper	93.41	-	-	-
park	98.90	-	-	-
planet	100.00	-	-	-
politicsblog	88.00	-	-	-
programminglanguage	-	-	81.32	-
recordlabel	55.56	50.00	-	-
religion	-	-	100.00	-
stadiumoreventvenue	96.70	-	-	-
televisionnetwork	97.80	-	-	-
televisionstation	87.78	-	-	-
vegetable	-	100.00	-	-
videogame	72.73	-	-	-
Average precision	90.61	90.44	76.85	100.00

Table 15: Performance of WebSets on NELL categories