Using Structured Text for Large-Scale Attribute Extraction

Sujith Ravi*
University of Southern California
Information Sciences Institute
Marina del Rey, California 90292
sravi@isi.edu

Marius Paşca
Google Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
mars@google.com

ABSTRACT

We propose a weakly-supervised approach for extracting class attributes from structured text available within Web documents. The overall precision of the extracted attributes is around 30% higher than with previous methods operating on Web documents. In addition to attribute extraction, this approach also automatically identifies values for a subset of the extracted class attributes.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.2.7 [**Artificial Intelligence**]: Natural Language Processing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Weakly-supervised information extraction, class attribute extraction, knowledge acquisition, structured text collections

1. INTRODUCTION

Information extraction systems typically exploit unstructured text within Web documents to acquire data for various applications. Despite the unstructured nature of documents available on the Web, some inherent structure often exists within the text. For example, a website listing telephone directory information contains recognizable fields such as a person's *name* and *phone number*. Web documents from encyclopedic resources such as Wikipedia [13] pertaining to specific entities (e.g., *Italy*) contain emphasized fields within the text (e.g., *capital, official languages, population, culture,* etc.) that signal important information regarding the entity. If the same set of fields appear within many documents related to multiple instances (e.g., *Italy, France, Japan,* etc.) representing the same class (e.g., *Country*), then these fields may be important properties of the particular class, and they may be potential candidates for attribute extraction. Sometimes, the text may also contain field tuples (e.g., ⟨*capital,Rome*⟩ for *Italy*), in which case they may provide information for extracting the value (e.g., *Rome*) of an attribute (e.g., *capital*) of a particular instance (e.g., *Italy*) [16].

Existing extraction systems use specific hand-crafted patterns [8] or schemas [6] to discover relational instances or structured tuples like ⟨*class, attribute*⟩ pairs from unstructured text. Most of these approaches are constrained to specific domains or particular target relations [3] specified in advance, and therefore do not gracefully scale to more generic classes and applications for other domains. A previous approach for attribute extraction from Web document collections looks at the immediate textual vicinity of instances (e.g., *Austria*) from a target class (e.g., *Country*) to identify potential attributes of the class [10]. However, the attributes that can be extracted using this technique are limited to specific types, following certain patterns (e.g., X-of-Y patterns [15]).

Sources such as Wikipedia or the CIA FactBook constitute appealing resources for obtaining documents related to specific instances of a class, and these often contain more than just attribute information. Some of these documents may provide additional information, such as relations between different entities associated with the particular instance. For example, the Wikipedia document pertaining to *Austria* also specifies that its *capital* is *Vienna*. Together, the pair ⟨*capital, Vienna*⟩ constitutes a piece of factual information for the instance *Austria* of the class *Country*. If we can find many such matching tuples for various instances of the class, then this would help us in mining facts related to the class on a large scale. But open-domain value extraction is a difficult problem, especially from noisy data like that available on the Web. There have been approaches in the past [3, 2] for extracting factual relations between objects, but many of them are geared towards extracting facts that correspond to answers to factual questions (e.g., *"What is the capital of Austria?"*), often following hand-crafted patterns. For any attribute extracted from a Web document relevant to a specific class instance, it is difficult to predict whether the source document also contains a matching value for the particular attribute. Even if the document contains

*Contributions made during an internship at Google.

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

Input:
- target class C, available as a set of instances $\{I\}$
- small set of seed phrases $\{K\}$ for the target class C expected in output

Output: Ranked list of attributes for C

Variables:
- $\{D\}$ = document collection for target class C
- $\{P\}$ = pool of candidate phrases
- $\{P_I\}$ = set of candidates $P \in \{P\}$ that were extracted for instance I
- $\{P_D\}$ = set of candidates $P \in \{P\}$ that were extracted from a specific document $D \in \{D\}$
- V_P = signature vector for candidate phrase P
- $\{V\}$ = Hierarchical pattern vector, one for every candidate phrase $P \in \{P\}$
- $SCORE(P)$ = Score for candidate phrase $P \in \{P\}$
- V_{ref} = Reference vector containing patterns from seed attributes

0. P = 0; {V} = nil; V_{ref} = nil
1. For each instance $I \in \{I\}$:
2. Query a search engine using the phrase I and add documents from top N search results to $\{D\}$
3. For each document $D \in \{D\}$
4. Scan D and select candidate phrases occurring within emphasized HTML tags, add these to $\{P_D\}$
5. For each candidate phrase $P \in \{P_D\}$
6. If ($\{P_D\} \cap \{K\}$!= nil) $\wedge$ (P is not a long phrase)
7. Add P to the candidate pool $\{P\}$
8. Retrieve corresponding HierarchicalPattern(P) from document D
9. V_P = Find/create entry for P in $\{V\}$ using pattern
10. For each candidate phrase $P \in \{P\}$
11. For each seed attribute $K \in \{K\}$
12. If ($P == K$)
13. V_P = Find entry for P in $\{V\}$
14. Merge elements (patterns) from V_P into reference vector V_{ref}
15. For each candidate phrase $P \in \{P\}$
16. V_P = Find entry for P in $\{V\}$
17. $SCORE(P)$ = Compute_Similarity(V_P, V_{ref})
18. Return Sorted list $\{P, SCORE(P)\}$

Figure 1: Weakly-Supervised Attribute Extraction Algorithm using Structured Text

this information, finding the actual value within the text is non-trivial.

In this paper, we present a new approach for extracting attributes from Web documents, making use of the existing structure (HTML tag hierarchy) within text. This method scales to a large number of classes, and the attribute extraction is also not restricted to any pre-determined pattern types (e.g., X-of-Y patterns). Instead, we automatically learn a wide variety of patterns for certain seed attributes of the target class provided as input, and use these to extract other attributes of the same class. The use of structured text for attribute extraction has another advantage: the weakly supervised algorithm used for attribute extraction can be extended to also perform value extraction, thus yielding ⟨attribute,value⟩ tuples for various class instances.

2. WEAKLY-SUPERVISED EXTRACTION FROM STRUCTURED TEXT

2.1 Attribute Extraction

The weakly supervised attribute extraction algorithm used in our approach is shown in Figure 1. Given a target class C, the algorithm retrieves documents from the Web that are relevant to instances representative of the class in Steps 1-2, then selects candidate attributes (in Steps 3-7) and corresponding patterns (in Steps 8-9) from the retrieved documents. Steps 10-14 are aimed at extracting patterns corresponding to a small set of seed attributes (5 known attributes per class) which guide the extraction of other attributes from the same class. The algorithm finally ranks the selected candidate attributes for the particular class (Steps 15-18).

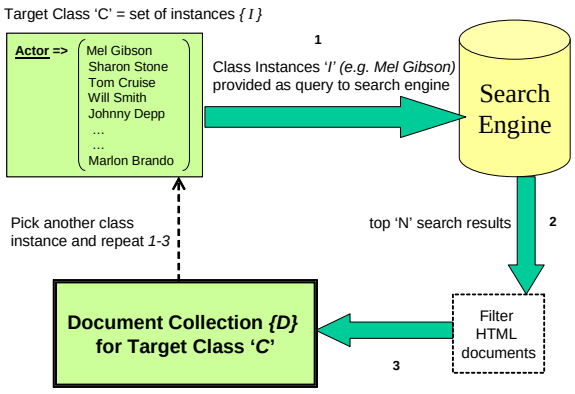

Figure 2: Fetching Web documents relevant to attribute extraction for a given class

2.1.1 Collecting Data from the Web

Each target class C is available as a set of representative entities or instances $\{I\}$. A large collection of instances (e.g., *Honda Accord, Audi A4, Mini Cooper, Ford Mustang* etc.) sharing the characteristic properties of a particular class (e.g., *CarModel*) can be mined from text collections available on the Web using various techniques that have been explored before [1, 14]. Given a particular class, a general-purpose search engine retrieves Web documents that are relevant to instances from the given class. In Steps 1-2 of the algorithm from Figure 1, each instance phrase I from the set representing the target class C is provided as a search query, and the top N documents returned for the query are fetched locally. Since we wish to exploit the structure within the document text, we limit our extraction to only documents containing HTML tag information; documents with extensions such as *.pdf, .doc, .txt*, etc. are ignored. Figure 2 illustrates via an example how relevant documents are collected from the Web to perform attribute extraction for a particular target class (*Actor*).

2.1.2 Selecting Candidate Attributes

Steps 3 and 4 in Figure 1 scan the retrieved documents and collect a pool of candidate phrases that could be potential attributes of the given class. For this purpose, we identify fields that are recognizable within a particular document, utilizing the structure information available in the form of HTML tags. For example, a document related to a particular entity (e.g., *Oracle*) from the given class (e.g., *Company*) might contain certain phrases (such as *ceo, stock price, headquarters, etc.*) which convey important information regarding the particular entity. These phrases are usually emphasized within the structured text using certain HTML tags such as *bold* (<b>), *italics* (<i>), *table columns* (<td>), *table headers* (<th>), *list elements* (<li>) etc. We extract such emphasized phrases from each document and add them to our pool of candidate attributes $\{P\}$. Figure 3 shows how the candidates *headquarters* and *net in-*

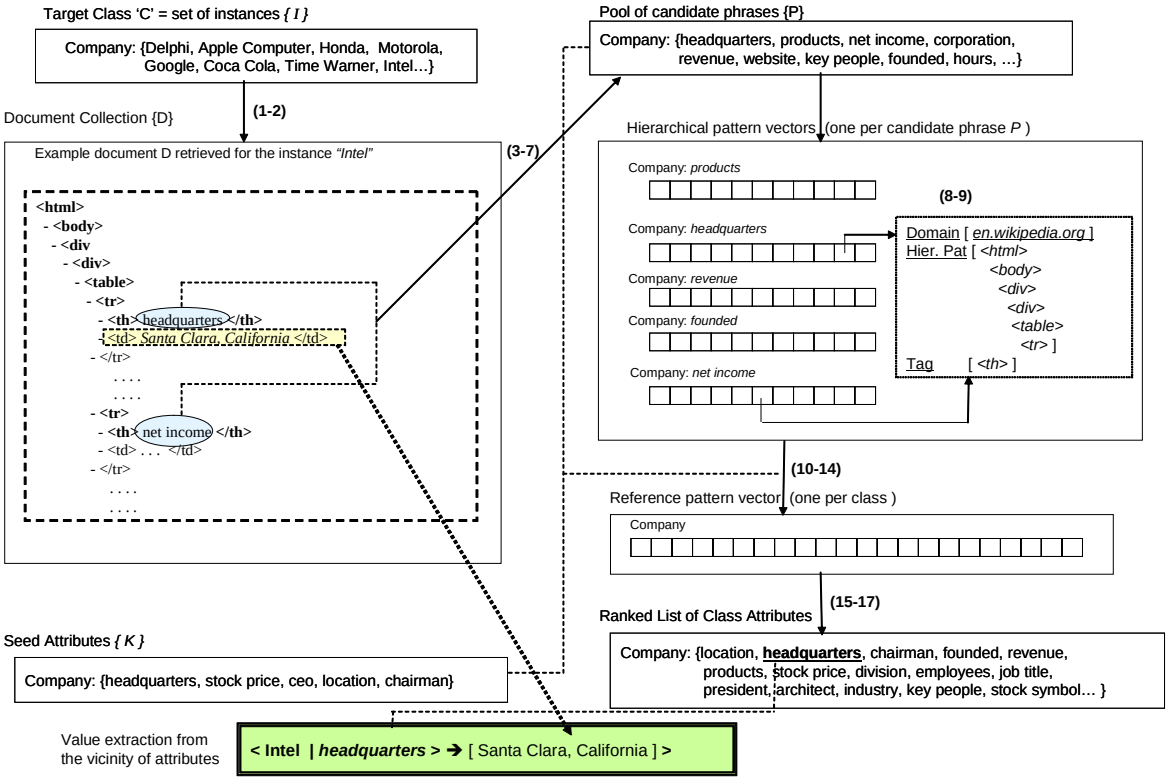

Figure 3: Data flow diagram for attribute extraction and value extraction

come (phrases enclosed within HTML tags <*th*>...</*th*>), are extracted from a Wikipedia document related to *Intel*, which is an instance of the class *Company*.

Filtering Out Spurious Candidates: Due to the ambiguous nature of some of the instance phrases, some of the retrieved documents may be irrelevant. For example, the instance *Chicago* is associated with multiple classes (*City* and *Movie*), and therefore attributes relevant for the class *Movie* might be added to the candidate pool for the class *City*, and vice versa. To avoid extracting a lot of such undesirable attributes from random documents, Step 6 filters out documents that do not contain fields matching at least one of the seed attributes in $\{K\}$. We also restrict our candidate selection to relatively short phrases, i.e., containing five words or less.

Extracting Hierarchical Patterns: For every candidate phrase generated, Step 8 of the algorithm from Figure 1 extracts the corresponding hierarchical tag pattern associated with it from the document. There have been many approaches in the past which utilize the HTML structure within documents for information extraction, but most of these approaches tend to focus only on specific elements like HTML tables within Web documents [4]. For a particular candidate P selected from document D, its hierarchical tag structure is composed of the HTML elements corresponding to its tag, parent tag, grandparent tag, and so on (all the way to the top of the document, within 10 previous tag levels). This tag structure along with other information (domain name of the website from which D was retrieved) constitutes a pattern. Figure 3 illustrates how the pattern corresponding to the candidate *headquarters* is extracted from a Wikipedia document for the instance *Intel*. All such patterns extracted for candidate P are then collected into a pattern vector V_P (Step 9 in Figure 1). The pattern vector represents the fingerprint or signature for a particular candidate phrase, and overlap between different pattern vectors indicates that candidates corresponding to these vectors share common patterns.

2.1.3 Ranking Class Attributes

Every candidate extracted for an instance I represents a potential attribute of the particular class C to which I belongs. In Figure 3, the candidates *headquarters* and *net income* extracted for the instance *Intel* are attributes of the class *Company*. The selection process builds a large pool of such candidate attributes for each target class. Ranking the attributes within the candidate pool can be done in multiple ways:

• *Frequency-based ranking:* One method is to rank the selected candidates for a particular class, by frequency [10] as shown in Equation 1. Each candidate P extracted for a class C is assigned a score based on the number of instances I from C that produced P. A candidate that is produced by more instances from the same class, receives a higher score.

$$SCORE_{freq}(P) = \frac{\mid I : ((I \in C) \wedge (I \; produces \; P)) \mid}{\mid I : (I \in C) \mid} \quad (1)$$

• *Hierarchical pattern-based ranking:* Pattern vectors V_P are populated for every phrase P in the candidate pool $\{P\}$, for a particular class C. The individual patterns within each vector are weighted elements (the frequency of occurrence within documents in the collection). Patterns from vectors corresponding to the candidate phrases that match any of the seed attributes (e.g., *headquarters*) are then aggregated in Steps 10-14 (Figure 1) to form a reference vector V_{ref} for

Class	Examples of Instances
Actor	Mel Gibson, Sharon Stone, Tom Cruise, Sophia Loren, Will Smith, Johnny Depp, Kate Hudson
AircraftModel	Boeing 747, Boeing 737, Airbus A380, Embraer 170, ATR 42, Boeing 777, Douglas DC 9, Dornier 228
Award	Nobel Prize, Pulitzer Prize, Webby Award, National Book Award, Prix Ars Electronica, Fields Medal
BasicFood	fish, turkey, rice, milk, chicken, cheese, eggs, corn, beans, ice cream
CarModel	Honda Accord, Audi A4, Subaru Impreza, Mini Cooper, Ford Mustang, Porsche 911, Chrysler Crossfire
CartoonCharacter	Mickey Mouse, Road Runner, Winnie the Pooh, Scooby-Doo, Homer Simpson, Bugs Bunny, Popeye
CellPhoneModel	Motorola Q, Nokia 6600, LG Chocolate, Motorola RAZR V3, Siemens SX1, Sony Ericsson P900
ChemicalElem	lead, silver, iron, carbon, mercury, copper, oxygen, aluminum, sodium, calcium
City	San Francisco, London, Boston, Ottawa, Dubai, Chicago, Amsterdam, Buenos Aires, Paris, Atlanta
Company	Adobe Systems, Macromedia, Apple Computer, Gateway, Target, Netscape, Intel, New York Times
Country	Canada, France, China, Germany, Australia, Lichtenstein, Spain, South Korea, Austria, Taiwan
Currency	Euro, Won, Lire, Pounds, Rand, US Dollars, Yen, Pesos, Kroner, Kuna
DigitalCamera	Nikon D70, Canon EOS 20D, Fujifilm FinePix S3 Pro, Sony Cybershot, Nikon D200, Pentax Optio 430
Disease	asthma, arthritis, hypertension, influenza, acne, malaria, leukemia, plague, tuberculosis, autism
Drug	Viagra, Phentermine, Vicodin, Lithium, Hydrocodone, Xanax, Vioxx, Tramadol, Allegra, Levitra
Empire	Roman Empire, British Empire, Ottoman Empire, Byzantine Empire, German Empire, Mughal Empire
Flower	Rose, Lotus, Iris, Lily, Violet, Daisy, Lavender, Magnolia, Tulip, Orchid
Holiday	Christmas, Halloween, Easter, Thanksgiving, Labor Day, Independence Day, Lent, Yule, Hanukkah
Hurricane	Hurricane Katrina, Hurricane Ivan, Hurricane Dennis, Hurricane Wilma, Hurricane Frances
Mountain	K2, Everest, Mont Blanc, Table Mountain, Etna, Mount Fuji, Mount Hood, Annapurna, Kilimanjaro
Movie	Star Wars, Die Hard, Air Force One, The Matrix, Lord of the Rings, Lost in Translation, Office Space
NationalPark	Great Smoky Mountains National Park, Grand Canyon National Park, Joshua Tree National Park
NbaTeam	Utah Jazz, Sacramento Kings, Chicago Bulls, Milwaukee Bucks, San Antonio Spurs, New Jersey Nets
Newspaper	New York Times, Le Monde, Washington Post, The Independent, Die Welt, Wall Street Journal
Painter	Leonardo da Vinci, Rembrandt, Andy Warhol, Vincent van Gogh, Marcel Duchamp, Frida Kahlo
ProgLanguage	A++, PHP, C, C++, BASIC, JavaScript, Java, Forth, Perl, Ada
Religion	Islam, Christianity, Voodoo, Buddhism, Judaism, Baptism, Taoism, Confucianism, Hinduism, Wicca
River	Nile, Mississippi River, Hudson River, Colorado River, Danube, Amazon River, Volga, Snake River
SearchEngine	Google, Lycos, Excite, AltaVista, Baidu, HotBot, Dogpile, WebCrawler, AlltheWeb, Clusty
SkyBody	Earth, Mercury, Saturn, Vega, Sirius, Polaris, Pluto, Uranus, Antares, Canopus
Skyscraper	Empire State Building, Sears Tower, Chrysler Building, Taipei 101, Burj Al Arab, Chase Tower
SoccerClub	Chelsea, Real Madrid, Juventus, FC Barcelona, AC Milan, Aston Villa, Real Sociedad, Bayern Munich
SportEvent	Tour de France, Super Bowl, US Open, Champions League, Bundesliga, Stanley Cup, FA Cup
Stadium	Stade de France, Olympic Stadium, Wembley Stadium, Soldier Field, Old Trafford, Camp Nou
TerroristGroup	Hezbollah, Khmer Rouge, Irish Republican Army, Shining Path, Tupac Amaru, Sendero Luminoso
Treaty	North Atlantic Treaty, Kyoto Protocol, Louisiana Purchase, Montreal Protocol, Berne Convention
University	University of Oslo, Stanford, CMU, Columbia University, Tsing Hua University, Cornell University
VideoGame	Half Life, Final Fantasy, Grand Theft Auto, Warcraft, Need for Speed, Metal Gear, Gran Turismo
Wine	Port, Champagne, Bordeaux, Rioja, Chardonnay, Merlot, Chianti, Cabernet Sauvignon, Pinot Noir
WorldWarBattle	D-Day, Battle of Britain, Battle of the Bulge, Battle of Midway, Battle of the Somme, Battle of Crete

Table 1: Target classes used for attribute extraction and examples of instances

the class (e.g., *Company*). Similarly to [9], the relevance of each candidate attribute (P) for the class is then computed as a similarity score of its pattern vector V_P with respect to the reference vector (V_{ref}) for the class (Steps 15-17 in Figure 1). The similarity score is obtained by computing the Jaccard coefficient between the two vectors as shown in Equation 2. A higher similarity score indicates that it is more likely for patterns within V_P to match those of the seed attribute patterns, which in turn suggests that the candidate phrase P could potentially be a good attribute for the given class.

$$SCORE_{hier}(P) = \frac{\mid V_P \cap V_{ref} \mid}{\mid V_P \cup V_{ref} \mid} \quad (2)$$

The framework described in Figure 1 (and illustrated via an example in Figure 3) is used to build two separate systems for attribute extraction: (1) using frequency-based ranking (which is the baseline for our comparison), and (2) using hierarchical pattern-based ranking. All the previous steps in attribute extraction (including candidate selection) are the same for both these systems. In addition to all the steps mentioned, the candidates may also pass through some post-processing steps to obtain the final ranked list of class attributes. Attributes that are added simultaneously to the candidate pools for multiple classes, are less useful in identifying characteristic properties that are unique to a particular class. These may include generic attributes like *history, definition,* etc. or phrases that commonly occur in boilerplate text within Web documents (e.g., *contents, contact information, help,* etc.). Consequently, if a candidate is extracted simultaneously for more than half the classes from the target class set, the candidate is removed from consideration.

2.2 Value Extraction as Part of Attribute Extraction

The approach presented above is generic and can be used to extract attributes for a large number of classes, with minimal supervision. We propose to leverage the attribute extraction framework (described in Section 2.1) to perform value extraction, using a simple extension. For a particular document D that was retrieved for class instance I, we find the most frequent pattern (HTML hierarchical tag pattern within the pattern vector V_P) corresponding to every attribute P that was extracted from D. This information is subsequently used for extracting matching values for P. For many class instances, structured text retrieved from encyclopedic sources contains many recognizable field tuples such as ⟨attribute,value⟩ pairs associated with the particu-

lar instance or entity described in the document. Usually, the fields constituting each such tuple occur in the vicinity of one other within the document. Therefore, documents containing attributes as fields might also contain potential matching values in nearby context.

Given an attribute P, the document D from which it was extracted, and the corresponding instance $I \in C$, the value extraction process collects the text VAL_P (from the tag element) immediately following the hierarchical tag pattern for P inside D, and returns the triplet $\langle I \mid P \Rightarrow VAL_P \rangle$ as factual information for the class instance (C: I). In Figure 3, for the instance *Intel* of the class *Company*, the text *"Santa Clara, California"* is matched as the value corresponding to the candidate attribute *headquarters*, by first locating the position of the attribute using the hierarchical pattern, and then selecting the immediately following element (in this case, $\langle td \rangle ... \langle /td \rangle$). Using this integrated approach, we can inexpensively identify matching values for some of the correctly identified attributes, if this information is present in the document. Some example values extracted in this manner are shown below.

Country: $\langle$ *Austria* | *population* $\Rightarrow$ 2007 *estimate* $= 8,316,487 \rangle$

Wine: $\langle$ *Pinot Gris* | *color* $\Rightarrow$ *White* $\rangle$

3. EXPERIMENTAL SETTING

3.1 Attribute Extraction

Target Classes: A total of 40 target classes [9] are provided as input in the form of sets of instances $\{I\}$ in the experiments. Table 1 illustrates target classes and example instances used in the attribute extraction experiments. The number of instances per class varies from 25 (for *SearchEngine*) to 1500 (for *Actor*). The classes also vary widely with respect to the domain or genre (e.g., Entertainment for *Movie*, Championship Leagues for *SportEvent*) and types of instances (e.g., Boeing or Airbus for *AircraftModel*) comprising the particular class.

Seed Attributes: As part of the weak supervision required for this approach, we provide a small set of 5 seed (known) attributes per class. These attributes are chosen independently, and are provided as input at the beginning of attribute extraction. An example of a seed attribute set is {*headquarters, stock price, ceo, location, chairman*} for the class *Company*.

Evaluation Methodology: Regardless of the differences in methodologies used for attribute extraction, all experiments produce a ranked list of candidate attributes for each of the 40 target classes. Multiple lists of attributes (one list per class) from all experiments are evaluated in the same manner using precision as the metric. To avoid any undesirable bias towards higher ranked attributes during evaluation, each list is sorted alphabetically into a merged list. Each attribute in this merged list is assigned a correctness label by a human judge, as shown in Table 2.

The strategy for assigning correctness labels is similar to the assessment method used in previous work [10]. An attribute is *"vital"* if it must be present in an ideal list of attributes for the target class; *"okay"* if it provides useful but non-essential information; and *"wrong"* if it is incorrect.

Label	Value	Examples of Attributes
vital	1.0	Actor: date of birth, Flower: botanical name
okay	0.5	Company: vision, NationalPark: reptiles
wrong	0.0	BasicFood: low carb, Movie: fiction

Table 2: Correctness labels for attribute quality assessment

All attributes extracted for each of the 40 target classes, are assigned correctness labels. These correctness labels are then mapped to quantitative values as shown in Table 2. The overall precision score at some rank N for a particular class C is then computed as the sum of the correctness values of the first N candidates in the ranked list of attributes extracted for C, divided by N.

Parameter Settings: There are a few parameters in the attribute extraction pipeline that can be used to tweak different aspects of the system. Varying one or more of these parameters may have a bearing on the final results (i.e., class attributes) generated by the system:

Top Documents [N = 50 or 200 (default=200)]: During data collection, we use a search engine to extract relevant documents (corresponding to top N search results) for member instances of the target class. We can vary the number of top results extracted per class instance by adjusting this parameter, and observe its effect on the attribute extraction results.

WordNet Pruning [WN = on/off (default=on)]: Since the candidate attributes are extracted from noisy Web data, there could be spurious entries that make it into the final ranked list of class attributes (e.g., *"1989", "3.2-mp",* etc.) During the post-processing stage in the pipeline, we can filter out candidates that do not contain meaningful English words or terms using a general-purpose lexical resource such as WordNet [7].

Comparative Attribute Extraction Runs: We compare three different systems for attribute extraction in our experiments:

(B) **- Baseline system with structured text:** The baseline system is implemented using the extraction framework shown in Figure 1, and uses the structured text within relevant Web documents (N = top 200 documents, per class instance) to select candidate attributes. The selected candidates are then ranked using frequency-based ranking (Equation 1 from Section 2.1.3). Generic attributes are removed and WordNet pruning is applied during the post-processing stage.

(S_{hier}) **- System using hierarchical patterns with structured text:** The second system is identical to the baseline system, except it uses the hierarchical tag patterns within the structured text to rank selected candidates (Equation 2 from Section 2.1.3). WordNet pruning and generic attribute removal strategies are similarly applied. This system is also capable of performing value extraction.

(D_{patt}) **- Previous approach with unstructured text:** We also implemented a third system using the attribute extraction strategy described in previous work [10]. This approach uses hand-crafted patterns (such as X-of-Y patterns) to extract attributes from unstructured text within Web documents. To ensure a faithful comparison, we used the same normalized ranking function introduced in [10] to rank the candidate attributes extracted for each class, since it was shown to perform better than a frequency-based ranking function.

Class	Precision																	
	@5			@10			@20			@30			@40			@50		
	B	D_{patt}	S_{hier}	B	D_{patt}	S_{hier}	B	D_{patt}	S_{hier}	B	D_{patt}	S_{hier}	B	D_{patt}	S_{hier}	B	D_{patt}	S_{hier}
Actor	0.60	0.30	1.00	0.40	0.45	0.90	0.43	0.55	0.83	0.42	0.56	0.83	0.36	0.55	0.74	0.30	0.55	0.68
BasicFood	0.20	0.40	1.00	0.60	0.25	0.90	0.40	0.27	0.85	0.33	0.20	0.72	0.30	0.15	0.58	0.27	0.16	0.46
Average-Class	0.53	0.57	0.87	0.48	0.54	0.76	0.44	0.50	0.65	0.40	0.47	0.59	0.38	0.44	0.55	0.36	0.42	0.52
Average-Class *(Imp)*	-	-	+53%	-	-	+41%	-	-	+30%	-	-	+26%	-	-	+25%	-	-	+24%
Average-Class *(Err)*	-	-	−70%	-	-	−48%	-	-	−30%	-	-	−23%	-	-	−20%	-	-	−17%

Table 4: Precision (at various ranks N) of attributes extracted by the three systems (B, D_{patt}, S_{hier}) for a couple of classes as well as average precision over all classes. The relative precision *Imp*rovement and *Err*or reduction (S_{hier} over D_{patt}) at various ranks, are also shown in the last two rows.

Class	Top Extracted Attributes
Actor	awards, height, career, age, birthplace, born, biography, quotes, birth name, personal life
AircraftModel	length, wingspan, range, wing area, crew, engines, cruise speed, empty weight, service ceiling
ChemicalElement	atomic number, symbol, atomic weight, melting point, density, electronic configuration, ionic radius
Country	population, climate, area, capital, currency, president, flag, economy, religions, ethnic groups
Hurricane	data, damage, stage, minimum sea-level pressure, action, deaths, hurricane, fatalities, peak gust
Painter	biography, paintings, nationality, bibliography, birthplace, works, exhibitions, born, died, prints

Table 3: Top attributes extracted (using system S_{hier}) from structured text in Web documents for a few target classes

The attributes extracted by all these systems are evaluated in the same manner, following the methodology described earlier in this section.

3.2 Value Extraction

The value extraction procedure runs simultaneously with attribute extraction, as described earlier in Figure 3. For every candidate attribute P extracted from text for a particular instance $I \in$ class C, we obtain some matching text VAL_P that is extracted as the value for P.

Target Classes: Since the evaluation of value extraction results is time consuming, we consider a smaller set of classes and instances as compared to earlier experiments - i.e., only 20 out of the 40 target classes that were used for attribute extraction are used for value evaluation. For each of these 20 classes, 5 instances are selected randomly from its member set to represent the class.

Attributes: The text within the extracted element VAL_P would constitute a meaningful value for the candidate P, only if P can be considered a valid attribute of the class C for which it was extracted. This is a minimum requirement for our value extraction approach to work. Consequently, from the ranked list of candidates extracted for a target class, we pick a set of 5 high quality attributes for that class, and use only these in our value evaluation experiments. For example, candidates such as *climate, currency, population, president,* and *religion* are considered high quality attributes for the class *Country*.

Evaluation Methodology: Each instance ($I \in C$) is combined with (1) an attribute P (one among the 5 attributes selected for class C), and (2) the value VAL_P extracted for the attribute P, from a document relevant to instance I; generating a triplet $\langle I \mid P \Rightarrow VAL_P \rangle$ for the class instance $(C : I)$. A total of 500 such triplets are generated for the 20 classes (5 attributes per class, 5 instances per class, 1 value per attribute-instance pair). Each of these triplets is manually evaluated by a human judge and assigned a correctness label. A triplet is assigned a label *"correct"* if it forms a valid tuple - i.e., if the real value matching attribute P for the class instance I exists within VAL_P; otherwise the triplet is labeled as *"incorrect"*. For example, the triplet $\langle$*Italy* $\mid$ *capital* $\Rightarrow$ *Rome*$\rangle$ is a valid tuple for the class *Country*. The precision scores for value extraction are then obtained by computing the number of triplets marked *correct*, divided by the total number of triplets.

4. EVALUATION RESULTS

4.1 Precision of Extracted Attributes

Every system used in the attribute extraction experiments produces a ranked list of class attributes corresponding to each target class. Table 3 shows some of the top attributes extracted for a few classes, using the seed-based approach with hierarchical tag patterns (S_{hier}). Many candidates ranked among the top list such as *awards, height, age, birthplace,* etc. for class *Actor*, and *atomic number, symbol, atomic weight,* etc. for class *ChemicalElement* represent vital characteristic properties of the particular class. But the list also contains a few wrong attributes, e.g., *stage, hurricane* for the class *Hurricane*, and *prints* for the class *Painter*. The list may also contain a few rather generic attributes, such as *career* or *personal life*, mixed with other more specific attributes of the same class (*Actor*).

Table 4 displays some of the results from our experiments with respect to precision scores for the top N attributes extracted by different systems. Individual precision scores for a couple of the target classes (*Actor, BasicFood*), as well as overall average precision (combining scores from all the target classes) for each system are shown for comparison.

It can be seen from Table 4 that the precision for different systems at a particular rank N varies with respect to the class. For some classes, like *Actor*, precision at lower ranks (N=5) is higher for the baseline system (B) when compared to the previous approach (D_{patt}), whereas for other classes like *BasicFood*, the phenomenon is reversed. However, the system S_{hier} introduced in this paper, which uses hierarchical tag patterns from structured text, consistently outperforms the other two systems (B and D_{patt}), both in terms of individual precision scores for most of the classes, as well as average precision for all the target classes, over a wide range of ranks. For instance, the system S_{hier} exhibits a 24% im-

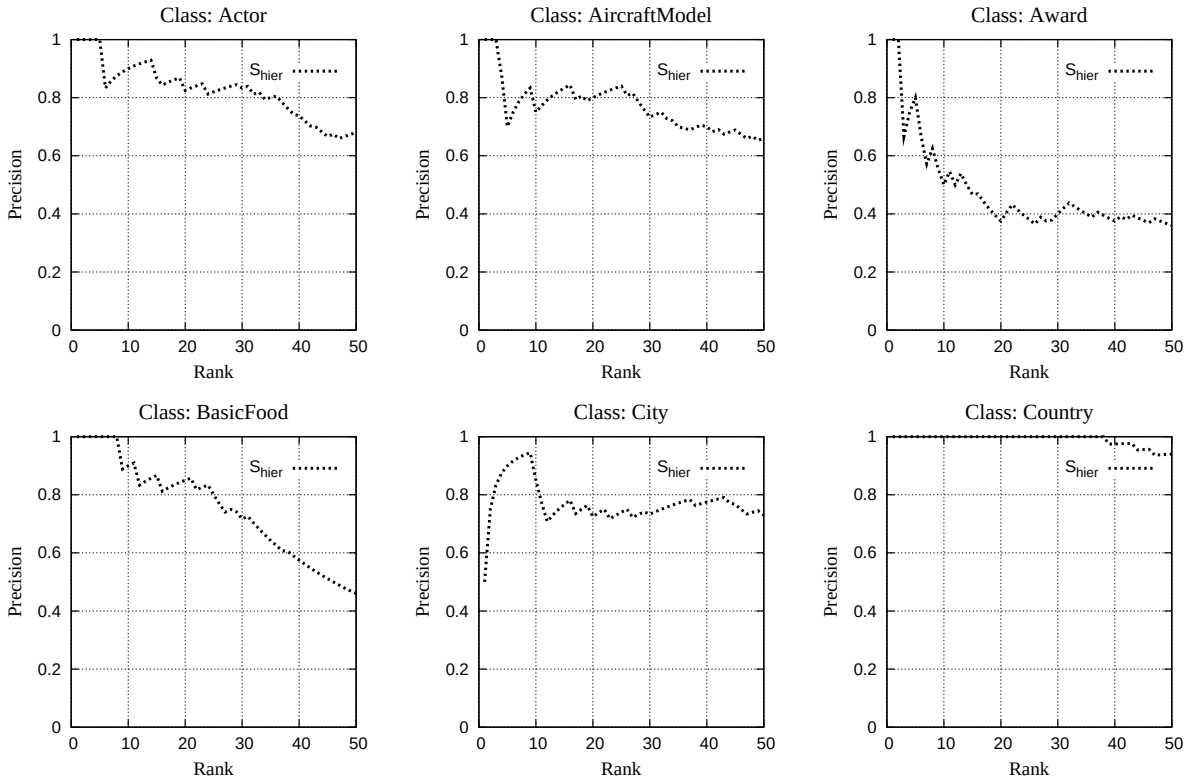

Figure 5: Individual precision scores at various ranks for attributes extracted by the system S_{hier} for a few target classes

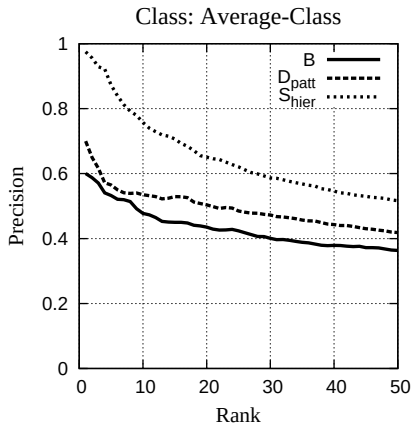

Figure 4: Relative system performance for attribute extraction in terms of average precision over all the classes

provement in precision (which maps to around 17% reduction in the error rate) at rank 50, when compared directly to results from the previous approach. Secondly, the results also indicate that ranking using hierarchical tag patterns (S_{hier}) produces much better attributes than a frequency-based ranking approach (baseline system B). The graph in Figure 4 further illustrates these results, by showing a head-to-head comparison of the different systems in terms of overall attribute precision at ranks 1 through 50. Figure 5 shows the individual precision results for some of the target classes. The quality of the extracted attributes varies from class to class. In general, the algorithm is able to extract good quality attributes for many of the classes (e.g. *Actor, AircraftModel, Country, etc.*) especially at lower ranks

Parameter	**Precision**				
	@10	@20	@30	@40	@50
Top Documents (N = 50)	0.76	0.65	0.56	0.53	0.49
Top Documents (N = 200)	0.76	0.65	0.58	0.52	0.49
WordNet pruning (WN = off)	0.76	0.65	0.58	0.52	0.49
WordNet pruning (WN = on)	0.76	0.65	0.59	0.55	0.52

Table 5: Effect of individually varying different parameters of the system S_{hier} on average precision of attributes

(N<30). For the class *Country*, the system yields high quality attributes, and the precision only starts to drop (from 1.0) at rank 40, whereas for the class *BasicFood*, the precision starts dropping at earlier ranks.

Varying parameter settings: Results for attribute extraction from a single system might also vary depending on the parameter settings used for the particular system. Table 5 shows that for the system S_{hier}, varying the parameter N (top search documents) does not affect the precision of extracted attributes by much, whereas applying WordNet pruning (WN=on) slightly improves the average precision at higher ranks.

Seed selection: The semi-supervised approach presented here uses seeds to mine patterns and extract attributes with similar patterns from Web documents, and hence varying the input seed attributes provided to the system might have an effect on the precision of extracted attributes. The attributes provided as seeds for attribute extraction were chosen randomly from a list of potential attributes compiled for each class, and may also include some generic ones, that do not describe any representative property of that particular class. Figure 6 shows how the average class precision of attributes is affected by providing the system with more

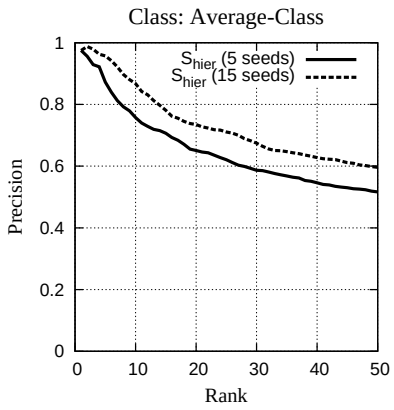

Figure 6: Effect of varying the number of *seeds per class* provided as input during attribute extraction, with respect to average precision over all the classes

input seeds. Using 15 seeds (instead of the original 5 seeds) per class helps in improving the average precision as shown in the graph, even at higher ranks (N=50). Adding more seeds allows the algorithm to extract more patterns for a particular class, and candidates simultaneously associated with many of these patterns are potentially good attributes for the class, thus improving the overall precision. However, even though more seeds might yield better performance, it is not always possible to find a large number of seeds, especially for uncommon classes. Since we wish to perform attribute extraction for a variety of classes, we limit the amount of supervision provided to the system, by limiting the number of seeds specified as input to five per class.

The quality of the attributes chosen as seeds might also influence the patterns (coverage, type) that are mined from Web documents for the target class, and as a result have an effect on the attributes that are extracted for the particular class. However, this paper does not focus on comparing the quality or type (generic/specific, etc.) of seeds provided as input, with the precision scores of extracted attributes.

4.2 Value Extraction Accuracy

Using the method proposed in this paper, values are extracted for a small set of high quality attributes corresponding to 20 different classes (5 instances per class, 5 attributes per class). Results from human evaluation (shown in Table 6) indicate that the system S_{hier} finds correct values for the provided instance-attribute pairs around 74% of the time. We acquire useful values for attributes related to classes such as *Country* and *Drug*; some *correct* examples are shown in Table 6, along with the class precision scores.

4.3 Discussion

The precision results confirm that structured text from the Web may be a good source for extracting information regarding relationships between objects and entities (as in the case of attribute extraction). A previous approach [10] using unstructured text extracts attributes from the immediate vicinity of class instances. In comparison, the algorithm presented in this paper is not only generic in nature (requiring minimal supervision); but also yields attributes that have higher quality (with a 30% improvement in precision). Despite the potential use of Web documents for attribute extraction, this data source also poses some disadvantages.

Class (C) / Precision	**Attribute (P)**	**Examples of class instances (I) for which correct values were found**
Actor *(0.68)*	awards	Adam Baldwin
	date of birth	Adam Baldwin, Allison Janney
Company *(0.84)*	ceo	Comcast, Halliburton
	headquarters	Comcast, Footstar
Country *(0.92)*	currency	Armenia, Austria
	president	Yemen, Mozambique
Drug *(0.92)*	adverse reactions	Atrovent, Levothyroxine
	mechanism of action	Atrovent, Effexor, Nitroquick
Average-Class Precision for Value Extraction = 0.74		

Table 6: Examples of high precision attributes corresponding to some class instances (i.e. $\langle I, P \rangle$ pairs), for which value extraction generated the *correct* answers

The large quantity of data available within Web documents may adversely cause a lot of spurious candidates to be extracted. Additionally, due to the limited availability of documents pertaining to certain classes, we might not be able to extract many attributes for the particular class. For example, many of the otherwise relevant documents retrieved for instances from the class *SearchEngine* do not contain much information describing the roles or characteristic features representative of the particular class. For classes such as *SearchEngine*, where fewer retrieved documents whose contents is rich in attributes may be available, attribute extraction using structured text within Web documents retrieved via search engines might not be feasible. Nevertheless, the use of structure does provide us with other advantages such as being able to find potential "values" for the candidate attributes that we extract. Using the hierarchical pattern and URL information obtained during attribute extraction, we can first locate the document from which the attribute was extracted, and then find potential matching values from its immediate vicinity or nearby context. This is a useful approach, leveraging the semi-supervised attribute extraction framework to guide value extraction, as a step towards mining factual information from Web data on a large scale.

5. RELATED WORK

Our extracted attributes are relations among objects in the given class, and objects or values from other classes. The lists of extracted attributes have direct benefits in gauging existing methods for harvesting pre-specified semantic relations [3, 11], towards the acquisition of relations that are of real-world interest to a wide set of Web users, e.g., towards finding *mechanisms of action* for a *Drug*.

In [5], the acquisition of attributes and other knowledge relies on users who explicitly specify it by hand. In contrast, we collect attributes automatically from structured text.

Several studies [15, 10, 17, 12] acquire attributes, possibly along with corresponding values, from Web documents. The method proposed in [15] applies handcrafted lexico-syntactic patterns to unstructured text within a small collection of Web documents. The evaluation consists in users manually assessing how natural the resulting candidate attributes are, when placed in a *wh-* question. In contrast to [15] and similarly to [10], our evaluation is stricter, since attributes that would otherwise pass the question answerability test used in [15] are marked as wrong in our evaluation.

In [10], the target classes are specified as sets of representative instances, which is similar to our method. However, the output attributes are collected by applying hand-written patterns (rather than using seed attributes) to unstructured text (rather than structured text).

The method introduced in [17] acquires attributes as well as associated values from structured text within Web documents, and does so by submitting queries to general-purpose search engines, and analyzing the contents of the top search results. There are, however, a few key differences. First, the search queries issued in [17] to identify relevant Web documents for a given class are based on hand-written query templates using the label of the target class, e.g., *"lists of movies"* for the class *Movie*. In comparison, our search queries correspond to a more flexible and scalable approach, in that they are issued automatically based on individual instances provided as input for each class. Besides using hand-written query templates, further manual intervention is employed in [17] in the form of strict hand-written patterns to filter out some of the spurious candidate collected from Web documents. In contrast, the method presented in our paper uniformly applies the same scoring function to all candidate attributes, and refrains from post-filtering candidate attributes with any hand-written patterns, with potentially higher coverage over a wider range of target classes.

6. CONCLUSION

Structured text within Web documents constitutes a useful source of information for the task of attribute extraction. A one-pass approach using structural patterns can be used to perform attribute and value extraction simultaneously. Using minimal supervision, this method extracts relevant attributes for a large number of classes spanning a wide range of domains. For some of the high precision attributes that are extracted, matching values can be found within the same document to yield pieces of factual information related to various class instances, which can then be used to build a fact repository or a similar type of knowledge resource.

7. REFERENCES

[1] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plaintext collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries (DL-00)*, pages 85–94, San Antonio, Texas, 2000.

[2] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2670–2676, Hyderabad, India, 2007.

[3] M. Cafarella, D. Downey, S. Soderland, and O. Etzioni. KnowItNow: Fast, scalable information extraction from the Web. In *Proceedings of the Human Language Technology Conference (HLT-EMNLP-05)*, pages 563–570, Vancouver, Canada, 2005.

[4] H. Chen, S. Tsai, and J. Tsai. Mining tables from large scale html texts. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*, pages 166–172, 2000.

[5] T. Chklovski and Y. Gil. An analysis of knowledge collected from volunteer contributors. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, pages 564–571, Pittsburgh, Pennsylvania, 2005.

[6] A. Doan, R. Ramakrishnan, F. Chen, P. DeRose, Y. Lee, R. McCann, M. Sayyadian, and W. Shen. Community information management. *IEEE Data Engineering Bulletin*, 29(1), 2006.

[7] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database and Some of its Applications*. MIT Press, 1998.

[8] T. Jayram, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Zhu. Avatar information extraction system. *IEEE Data Engineering Bulletin*, 29(1), 2006.

[9] M. Paşca. Organizing and searching the World Wide Web of facts - step two: Harnessing the wisdom of the crowds. In *Proceedings of the 16th World Wide Web Conference (WWW-07)*, pages 101–110, Banff, Canada, 2007.

[10] M. Paşca, B. Van Durme, and N. Garera. The role of documents vs. queries in extracting class attributes from text. In *Proceedings of the 16th International Conference on Information and Knowledge Management (CIKM-07)*, pages 485–494, Lisbon, Portugal, 2007.

[11] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 113–120, Sydney, Australia, 2006.

[12] K. Probst, R. Ghani, M. Krema, A. Fano, and Y. Liu. Semi-supervised learning of attribute-value pairs from product descriptions. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2838–2843, Hyderabad, India, 2007.

[13] M. Remy. Wikipedia: The free encyclopedia. *Online Information Review*, 26(6):434, 2002.

[14] K. Shinzato and K. Torisawa. Acquiring hyponymy relations from Web documents. In *Proceedings of the 2004 Human Language Technology Conference (HLT-NAACL-04)*, pages 73–80, Boston, Massachusetts, 2004.

[15] K. Tokunaga, J. Kazama, and K. Torisawa. Automatic discovery of attribute words from Web documents. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05)*, pages 106–118, Jeju Island, Korea, 2005.

[16] F. Wu and D. Weld. Automatically refining the Wikipedia infobox ontology. In *Proceedings of the 17th World Wide Web Conference (WWW-08)*, pages 635–644, Beijing. China, 2008.

[17] N. Yoshinaga and K. Torisawa. Open-domain attribute-value acquisition from semi-structured texts. In *Proceedings of the 6th International Semantic Web Conference (ISWC-07), Workshop on Text to Knowledge: The Lexicon/Ontology Interface (OntoLex-2007)*, pages 55–66, Busan, South Korea, 2007.