Organizing and Searching the World Wide Web of Facts - Step Two: Harnessing the Wisdom of the Crowds

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

ABSTRACT

As part of a large effort to acquire large repositories of facts from unstructured text on the Web, a seed-based framework for textual information extraction allows for weakly supervised extraction of class attributes (e.g., *side effects* and *generic equivalent* for drugs) from anonymized query logs. The extraction is guided by a small set of seed attributes, without any need for handcrafted extraction patterns or further domain-specific knowledge. The attributes of classes pertaining to various domains of interest to Web search users have accuracy levels significantly exceeding current state of the art. Inherently noisy search queries are shown to be a highly valuable, albeit unexplored, resource for Web-based information extraction, in particular for the task of class attribute extraction.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.2.7 [**Artificial Intelligence**]: Natural Language Processing; I.2.6 [**Artificial Intelligence**]: Learning; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Knowledge acquisition, class attributes, named entities, fact extraction, Web search queries, unstructured text

1. INTRODUCTION

1.1 Background

Although the information in large textual collections such as the Web is available in the form of individual textual documents, the human knowledge encoded within the documents can be seen as a hidden, implicit Web of classes of objects (e.g., named entities), interconnected by relations applying to those objects (e.g., facts). The acquisition of an extensive World Wide Web of facts from textual documents is an effort to improve Web search [12] that also fits into the far-reaching goal of automatically constructing knowledge bases from unstructured text [17].

Recent work on large-scale information extraction holds much promise, as it scales well to Web-sized text collections through to an emphasis on lightweight methods for extracting facts of a pre-defined type (e.g., InstanceOf [15], Person-AuthorOf-Invention [10] or Country-CapitalOf-City [2]). Beyond algorithmic differences and choice of underlying resources, these methods take as input a small set of facts of a pre-specified type, and mine a textual document collection to acquire many other facts of the same type. The resulting functionality is an excellent first step towards extracting the World Wide Web of facts:

• Step One (mining from Web documents, as described in [12]): for a target fact type (e.g., birth years of people), starting from as few as 10 seed facts such as (*John Lennon*, *1941*), mine a collection of textual Web documents to acquire sets in the order of a million facts of the same type.

To fully take advantage of this first step, however, one would need to identify the types of facts or class attributes that are of common interest to people in general, and to Web search users in particular:

• Step Two (mining from query logs, as introduced in this paper): for a target class (e.g., *Drug* or *AircraftModel*), starting from as few as 5 seed attributes (e.g., *side effects* and *maximum dose* for *Drug*, or *seating arrangement* and *wingspan* for *AircraftModel*) and/or 10 seed instances (e.g., *Vicodin* and *Xanax* for *Drug*, or *Boeing 747* and *Airbus 380* for *AircraftModel*), mine a collection of Web search queries to acquire large sets of attributes for the same class.

1.2 Contributions

The seed-based identification of prominent class attributes from unstructured text, without any further domain knowledge, corresponds to a second, more general step towards building the World Wide Web of facts. In this light, the main contributions of this paper are twofold:

1) We introduce a weakly supervised framework in Section 2.2, for mining Web *search queries* in order to *explicitly* extract open-domain knowledge that is expected to be meaningful and suitable for later use. In contrast, previous work that looks at query logs as a useful resource does so only to *implicitly* derive signals improving the quality of various tasks such as information retrieval, whether through re-ranking of the retrieved documents [22], query expansion [4], or the development of spelling correction models [7]. Conversely, previous studies in large-scale information ex-

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

traction uniformly choose to capitalize on document collections [11] rather than queries as preferred data source, thus failing to take advantage of the wisdom of the (search) crowds, to which millions of Web users contribute daily.

2) We illustrate how the proposed generic extraction framework applies to the concrete task of class attribute extraction in Section 2.3. To properly ensure varied experimentation on several dimensions, the evaluation in Section 3 computes the precision of attributes extracted for as many as 40 different target classes (*CarModel*, *City*, *Drug*, *VideoGame* etc.), chosen liberally from a wide range of domains of interest. As an illustration of the scope and time-intensive nature of the evaluation, one of its pre-requisites is the manual assessment of the correctness of more than 18,000 candidate attributes. The precision numbers over the target classes are excellent both in absolute value (0.90 for prec@10, 0.85 for prec@20, and 0.76 for prec@50), and relatively to the quality of attributes extracted with handcrafted patterns from query logs (with precision increasing by 25% for prec@10, 32% for prec@20, and 43% for prec@50).

1.3 Potential Applications

Besides their intended role in assembling a high-coverage World Wide Web of facts, the extracted class attributes have an array of other applications. In Web publishing, the attributes constitute topics (e.g., *radius*, *surface gravity*, *orbital velocity*) to be suggested automatically, as human contributors manually add new entries (e.g., for a newly discovered celestial body) to resources such as Wikipedia [16]. In open-domain question answering, the attributes are useful in expanding and calibrating existing answer type hierarchies [8] towards frequent information needs. In Web search, the results returned to a query that refers to a named entity (e.g., *Pink Floyd*) can be augmented with a compilation of specific facts, based on the set of attributes extracted in advance for the class to which the named entity belongs. Moreover, the original query can be refined into semantically-justified query suggestions, by concatenating it with one of the top extracted attributes for the corresponding class (e.g., *Pink Floyd albums* for *Pink Floyd*).

Attribute extraction is a powerful tool in building new search verticals in Web search semi-automatically, for example to improve or provide alternative views of search results for popular topics such as health, travel and so forth.

2. EXTRACTION FROM SEARCH QUERIES

2.1 Mining Queries Rather Than Documents

From a quantitative standpoint, the amount of text within query logs is at a clear disadvantage against the much larger textual content available within document collections such as the Web. When compared to a query, which contains only two words on average, a textual document is orders of magnitude larger. The size difference is exacerbated in very large document collections, with the leading Web search engines currently providing access to billions of documents. At least in theory, this has implications on the potential quality of the extracted information, since more data usually means better results. Indeed, given enough documents, inexpensive algorithms produce results sometimes rivaling those output by more complex methods [5].

A large percentage of Web queries suffer from ambiguity as a result of underspecified information needs, limited or no grammatical structure due to the use of keywords rather than natural language, and frequent typos and misspellings due to the hurried pace at which Web users typically enter queries. Even though ambiguity is a notorious issue in sentence processing, when compared to search queries the content found within a document is relatively clear. As opposed to the casual typing of search queries, authors of documents tend to pay more attention to both form and content, to pass their message across to readers through coherent sentences in natural language. While this is particularly true for genres such as news or scientific articles, it also applies to other less formal texts such as Web documents.

Since major Web search engines do not currently support truly interactive search sessions, each query is a self-contained request for information. While documents are less ambiguous than queries due to the size restrictions on the respective mediums, most search queries are in fact typed with this issue in mind. Many Web users have learned to use the most meaningful and least ambiguous terms available when searching for information.

Perhaps the most intriguing aspect of queries is, however, their ability to indirectly capture human knowledge, precisely as they inquire about what is already known. Indeed, users formulate their queries based on the commonsense knowledge that they already possess at the time of the search. Search queries play two roles simultaneously: in addition to requesting new information, they also indirectly convey knowledge in the process. If knowledge is generally prominent or relevant, people will eventually ask about it, especially as the number of users and the quantity and breadth of the available knowledge increase, as it is the case with the Web as a whole. Query logs convey knowledge through requests that may be answered by the knowledge asserted in expository text of document collections.

2.2 Weakly Supervised Extraction Framework

Figure 1 describes the proposed algorithm for information extraction from anonymized search queries. The target class $\mathcal{C}$ (e.g., *VideoGame*), for which a certain type of phrases (e.g., class attributes) need to be extracted from query logs, is available in the form of a set of representative instances $\mathcal{I}$ (e.g., *Grand Theft Auto*, *Street Fighter II* and *Age of Empires*). As opposed to highly supervised methods that rely on handcrafted patterns to extract information from text [13], the knowledge guiding our extraction framework is limited to a small set of seed phrases $\mathcal{K}$ (e.g., *price*, *creator* and *genre*) that are known to be part of the desired output (e.g., attributes) for the class $\mathcal{C}$ (e.g., *VideoGame*).

Since a class (concept) is traditionally a placeholder for a set of instances (objects) that share similar properties [13], the desired phrases (of interest) $\mathcal{P}$ for a given class $\mathcal{C}$ can be derived by extracting and merging candidate phrases for individual instances $\mathcal{I}$ of the class. Step 1 in Figure 1 exploits the instances $\mathcal{I}$ to collect a large pool of noisy (high recall, low precision) candidate phrases $\mathcal{P}$ that are associated to various instances, and therefore are associated to the class. Any method may be used to collect the pool of candidate phrases $\mathcal{P}$, as long as most of the seed phrases $\mathcal{K}$ are likely to be in that pool. As a brute-force example, the set of n-grams that occur together with one of the instances in any of the search queries is a catch-all (although extremely noisy) pool of candidate phrases.

Steps 2 through 9 in Figure 1 finds queries that contain

Input: target class $\mathcal{C}$, available as a set of instances $\{\mathcal{I}\}$
. small set of seed phrases $\{\mathcal{K}\}$ expected in output
. large repository of search queries $\{\mathcal{Q}\}$
Output: ranked list of phrases for $\mathcal{C}$
Variables: $\{\mathcal{P}\}$ = pool of candidate phrases
. $\mathcal{T}_Q$ = query template
. $\mathcal{F}_Q$ = query frequency in logs
. $\mathcal{V}_P$ = search signature vector
. $\{\mathcal{V}\}$ = search-signature vectors, one per $\mathcal{P}$
. $\{\mathcal{S}\}$ = vector of scores, one score per $\mathcal{P}$
. $\mathcal{V}_K$ = reference search-signature vector
Steps:
00. $\{\mathcal{P}\} = \emptyset$; $\mathcal{T}$ = nil; $\mathcal{F}_Q = 0$; $\{\mathcal{V}\} = \emptyset$; $\mathcal{V}_K$ = nil
01. Collect $\{\mathcal{P}\}$ from $\{\mathcal{Q}\}$ based on $\{\mathcal{I}\}$
02. For each query $\mathcal{Q}$ in $\{\mathcal{Q}\}$
03. For each candidate phrase $\mathcal{P}$ in $\{\mathcal{P}\}$
04. For each instance $\mathcal{I}$ in $\{\mathcal{I}\}$
05. If ($\mathcal{Q}$ contains both $\mathcal{P}$ and $\mathcal{I}$)
06. $\mathcal{T}_Q$ = QueryRemainderTemplate($\mathcal{Q}$, $\mathcal{P}$, $\mathcal{I}$)
07. $\mathcal{F}_Q$ = QueryFrequency($\mathcal{Q}$)
08. $\mathcal{V}_P$ = Find/create entry for $\mathcal{P}$ in $\{\mathcal{V}\}$
09. Update weight of $\mathcal{T}_Q$ in $\mathcal{V}_P$ based on $\mathcal{F}_Q$
10. For each candidate phrase $\mathcal{P}$ in $\{\mathcal{P}\}$
11. For each seed phrase $\mathcal{K}$ in $\{\mathcal{K}\}$
12. If ($\mathcal{P}$ == $\mathcal{K}$)
13. $\mathcal{V}_P$ = Find entry for $\mathcal{P}$ in $\{\mathcal{V}\}$
14. If ($\mathcal{V}_P$!= nil)
15. Merge elements of $\mathcal{V}_P$ into $\mathcal{V}_K$
16. For each candidate phrase $\mathcal{P}$ in $\{\mathcal{P}\}$
17. $\mathcal{V}_P$ = Find entry for $\mathcal{P}$ in $\{\mathcal{V}\}$
18. $\mathcal{S}$.At($\mathcal{P}$) = ComputeSimScore($\mathcal{V}_P$, $\mathcal{V}_K$)
19. Return SortedList($\{\mathcal{P}\}$, $\{\mathcal{S}\}$)

Figure 1: Generic Framework for Weakly Supervised Information Extraction from Queries

both an instance $\mathcal{I}$, and a candidate phrase $\mathcal{P}$. The remainder of a matching query, that is, the concatenation of the remaining prefix, infix and postfix (any of which may be empty), becomes an entry in a query template vector that acts as a *search signature* of the candidate phrase with respect to the class. For example, given the instance *Intel* in the class *Company* and the candidate phrases *headquarters* and *mission statement*, the queries *"where is the world headquarters for intel corporation"* and *"mission statement for intel"* respectively produce the templates:

- $[\textit{where is the world}]_{prefix}$ $[\textit{for}]_{infix}$ $[\textit{corporation}]_{postfix}$;
- $[\ ]_{prefix}$ $[\textit{for}]_{infix}$ $[\ ]_{postfix}$.

Thus, query templates are added as weighted elements in the search-signature vector of each candidate phrase. The weights aggregate the frequency-based contribution of distinct queries (via distinct instances) to the same template (e.g., as another query may also ask about the world headquarters but for *Oracle*, rather than *Intel*). Steps 10 through 15 in Figure 1 introduce weak supervision in the extraction process. They identify the vectors associated with the seed phrases $\mathcal{K}$ that are known a-priori to be part of the desired output. Those vectors are merged into a reference search-signature vector that can be thought of as a loose search fingerprint of the desired output with respect to the class. In steps 16 through 19, the similarity scores among the search-signature vector of each candidate phrase, on one hand, and the reference search-signature vector, on the other hand, induce a ranking over the candidate phrases and determine the list of candidate phrases returned as output.

2.3 Class Attribute Extraction

An immediate application of the proposed extraction framework is the extraction of class attributes. Specifically, given a set of target classes specified as sets of representative instances, and a set of seed attributes for each class, the goal is to extract relevant class attributes from query logs, without relying on any further domain knowledge.

Figure 2 illustrates the extraction of class attributes from queries. The numbered arrows correspond to steps from the generic algorithm described earlier in Figure 1. As shown in the upper part of Figure 2, it is straightforward to collect a very large (and extremely noisy) pool of candidate attributes, by identifying the queries which contain one of the class instances (e.g., *delphi* and *apple computer*) at one extremity (e.g., *"installing delphi"* and *"apple computer headquarters"*), and collecting the remainders of the queries as candidate attributes (e.g., *installing* and *headquarters* for the class *Company*).

The search-signature vectors are populated for each candidate attribute in a second pass over the query logs. For instance, the query *"installing oracle 8.1-7 on solaris 8"*, containing both a class instance (*Oracle*) and a candidate attribute (*installing*) produces an entry in the search-signature vector of *installing* with respect to the class *Company*. The entry corresponds to the unique query template $[\ ]_{prefix}$ $[\ ]_{infix}$ $[\textit{8.1-7 on solaris 8}]_{postfix}$. Similarly, the query *"coca cola company one year stock price target"* results in a new entry being added to the search-signature vector of *stock price* with respect to the same class *Company*, corresponding to the query template $[\ ]_{prefix}$ $[\textit{company one year}]_{infix}$ $[\textit{target}]_{postfix}$. After combining the vectors associated with the seed attributes (*stock price*, *headquarters* etc.) into a reference vector for the class, the relevance of each candidate attribute for the class is computed as the similarity score of the vector associated to the candidate attribute, with respect to the reference vector for the class.

3. EVALUATION

3.1 Experimental Setting

Data: The input to the experiments is a random sample of around 50 million unique, fully-anonymized queries in English submitted by Web users to the Google search engine in 2006. All queries are considered independently of one another, whether they were submitted by the same user or different users, within the same or different search sessions.

Figure 3 shows the distribution of the queries from the random sample, according to the number of words in each query. If multiple occurrences of identical queries count towards the computation of the distribution, a fraction of 14.5%, 31.3%, 26.5%, 13.2% and 7.5% of the queries from the random sample contain 1, 2, 3, 4 and 5 words respectively, as shown by the solid line in Figure 3. Put differently, 93% of the queries from the sample contain 5 words or less. Only 0.7% of the queries consist of more than 10 words. If the computation ignores the frequency in the logs of each query, which corresponds to the dotted line in the figure, the distribution moderately shifts to indicate longer unique queries on average. Only 2.4% of the unique queries contain 1 word, whereas 14.3%, 27.8%, 22.9% and 14.9%

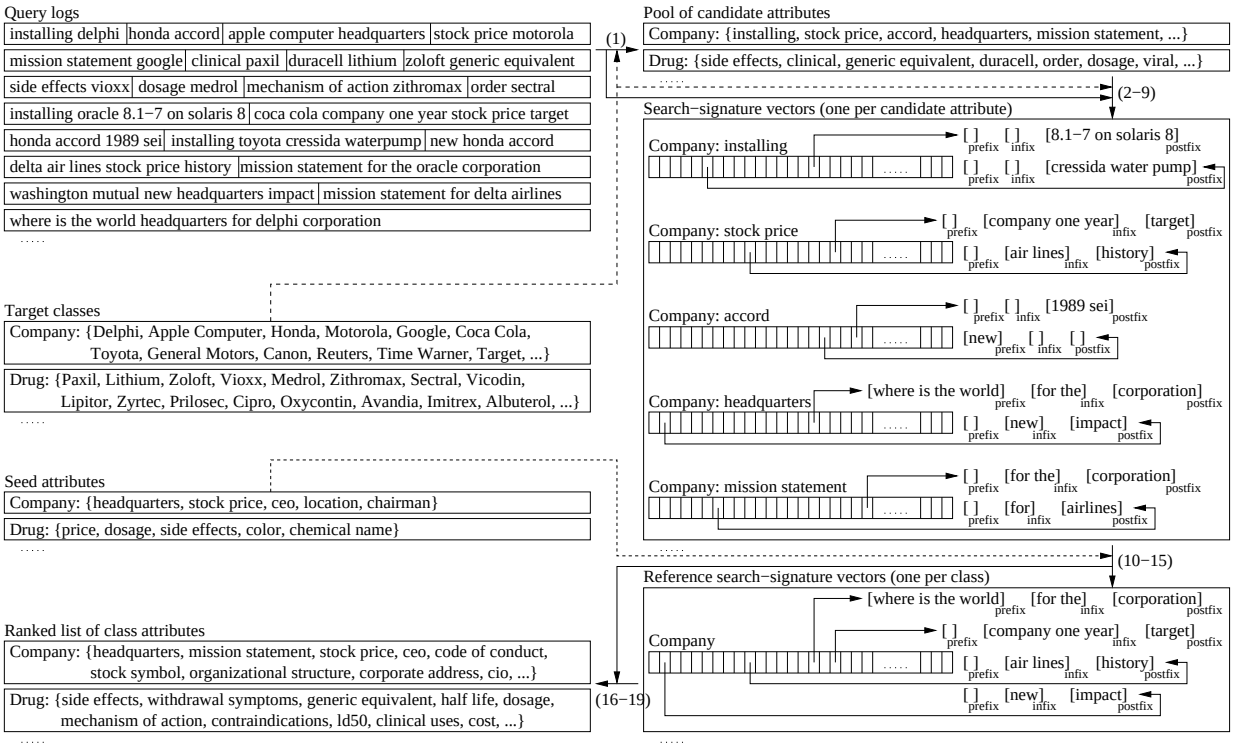

Figure 2: Overview of weakly supervised extraction of attributes from query logs

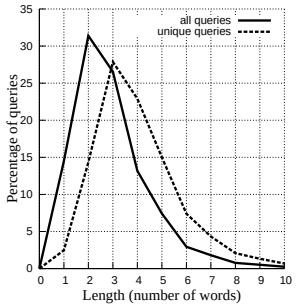

Figure 3: Percentage of input queries of various lengths, computed over all queries (including duplicates) and over unique queries

contain 2, 3, 4 and 5 words respectively. This corresponds to 82.5% unique queries with 5 words or less; 1.6% contain more than 10 words. The query length peaks at 3 words for unique queries, as compared to 2 words when considering all queries.

Despite the differences in the distributions of unique vs. all queries, Figure 3 confirms that most search queries, which constitute the input data to the experiments, are relatively short. Therefore, the amount of input data that is actually usable by the extraction method is only a fraction of the available 50 million queries, since an attribute cannot be extracted for a given class unless it occurs together with a class instance in an input query, which is a condition that is less likely to be satisfied in the case of short queries.

Target Classes: The target classes selected for experiments are each specified as an (incomplete) set of representative instances, details on which are given in Table 1. The number of instances varies from 25 (for *SearchEngine*) to 1500 (for *Actor*), with a median of 172 instances per class. The classes also differ with respect to the domain of interest (e.g., Health for *Drug* vs. Entertainment for *Movie*), instance capitalization (e.g., instances in *BasicFood* usually occur in text in lower rather than upper case), and conceptual type (e.g., abstraction for *Religion* vs. group for *SoccerClub* vs. activity for *VideoGame*). Therefore, we choose what we feel to be a large enough number of classes (40) to properly ensure varied experimentation on several dimensions, while taking into account the time intensive nature of manual accuracy judgments often required in the evaluation of information extraction systems [2].

Seed Attributes: Besides the set of its representative instances, each target class is accompanied by 5 seed attributes. For an objective evaluation, the seed attributes are chosen independently of whether it is possible to extract them from the collection of search queries with the proposed method, and without checking whether they actually occur within the search queries. Examples of complete seed attribute sets are {*quality*, *speed*, *number of users*, *market share*, *reliability*} for *SearchEngine*; {*symbol*, *atomic number*, *discovery date*, *mass*, *classification*} for *ChemicalElem*; and {*dean*, *number of students*, *research areas*, *alumni*, *mascot*} for *University*.

Similarity Functions: As described earlier, the relevance of a candidate attribute for a class is computed in Step 18 of Figure 1 as a similarity score between the search-signature vector associated to the candidate phrase (i.e., attribute), on one hand, and the reference search-signature vector for the class, on the other hand. Rather than arbitrarily choosing the similarity function driving the computation, we prefer to compare several similarity functions used the

Class (Size)	Examples of Instances
Actor (1500)	Mel Gibson, Sharon Stone, Tom Cruise, Sophia Loren, Will Smith, Johnny Depp, Kate Hudson
AircraftModel (217)	Boeing 747, Boeing 737, Airbus A380, Embraer 170, ATR 42, Boeing 777, Douglas DC 9, Dornier 228
Award (200)	Nobel Prize, Pulitzer Prize, Webby Award, National Book Award, Prix Ars Electronica, Fields Medal
BasicFood (155)	fish, turkey, rice, milk, chicken, cheese, eggs, corn, beans, ice cream
CarModel (368)	Honda Accord, Audi A4, Subaru Impreza, Mini Cooper, Ford Mustang, Porsche 911, Chrysler Crossfire
CartoonChar (50)	Mickey Mouse, Road Runner, Winnie the Pooh, Scooby-Doo, Homer Simpson, Bugs Bunny, Popeye
CellPhoneModel (204)	Motorola Q, Nokia 6600, LG Chocolate, Motorola RAZR V3, Siemens SX1, Sony Ericsson P900
ChemicalElem (118)	lead, silver, iron, carbon, mercury, copper, oxygen, aluminum, sodium, calcium
City (589)	San Francisco, London, Boston, Ottawa, Dubai, Chicago, Amsterdam, Buenos Aires, Paris, Atlanta
Company (738)	Adobe Systems, Macromedia, Apple Computer, Gateway, Target, Netscape, Intel, New York Times
Country (197)	Canada, France, China, Germany, Australia, Lichtenstein, Spain, South Korea, Austria, Taiwan
Currency (55)	Euro, Won, Lire, Pounds, Rand, US Dollars, Yen, Pesos, Kroner, Kuna
DigitalCamera (534)	Nikon D70, Canon EOS 20D, Fujifilm FinePix S3 Pro, Sony Cybershot, Nikon D200, Pentax Optio 430
Disease (209)	asthma, arthritis, hypertension, influenza, acne, malaria, leukemia, plague, tuberculosis, autism
Drug (345)	Viagra, Phentermine, Vicodin, Lithium, Hydrocodone, Xanax, Vioxx, Tramadol, Allegra, Levitra
Empire (78)	Roman Empire, British Empire, Ottoman Empire, Byzantine Empire, German Empire, Mughal Empire
Flower (59)	Rose, Lotus, Iris, Lily, Violet, Daisy, Lavender, Magnolia, Tulip, Orchid
Holiday (82)	Christmas, Halloween, Easter, Thanksgiving, Labor Day, Independence Day, Lent, Yule, Hanukkah
Hurricane (74)	Hurricane Katrina, Hurricane Ivan, Hurricane Dennis, Hurricane Wilma, Hurricane Frances
Mountain (245)	K2, Everest, Mont Blanc, Table Mountain, Etna, Mount Fuji, Mount Hood, Annapurna, Kilimanjaro
Movie (626)	Star Wars, Die Hard, Air Force One, The Matrix, Lord of the Rings, Lost in Translation, Office Space
NationalPark (59)	Great Smoky Mountains National Park, Grand Canyon National Park, Joshua Tree National Park
NbaTeam (30)	Utah Jazz, Sacramento Kings, Chicago Bulls, Milwaukee Bucks, San Antonio Spurs, New Jersey Nets
Newspaper (599)	New York Times, Le Monde, Washington Post, The Independent, Die Welt, Wall Street Journal
Painter (1011)	Leonardo da Vinci, Rembrandt, Andy Warhol, Vincent van Gogh, Marcel Duchamp, Frida Kahlo
ProgLanguage (101)	A++, PHP, C, C++, BASIC, JavaScript, Java, Forth, Perl, Ada
Religion (128)	Islam, Christianity, Voodoo, Buddhism, Judaism, Baptism, Taoism, Confucianism, Hinduism, Wicca
River (167)	Nile, Mississippi River, Hudson River, Colorado River, Danube, Amazon River, Volga, Snake River
SearchEngine (25)	Google, Lycos, Excite, AltaVista, Baidu, HotBot, Dogpile, WebCrawler, AlltheWeb, Clusty
SkyBody (97)	Earth, Mercury, Saturn, Vega, Sirius, Polaris, Pluto, Uranus, Antares, Canopus
Skyscraper (172)	Empire State Building, Sears Tower, Chrysler Building, Taipei 101, Burj Al Arab, Chase Tower
SoccerClub (116)	Chelsea, Real Madrid, Juventus, FC Barcelona, AC Milan, Aston Villa, Real Sociedad, Bayern Munich
SportEvent (143)	Tour de France, Super Bowl, US Open, Champions League, Bundesliga, Stanley Cup, FA Cup
Stadium (190)	Stade de France, Olympic Stadium, Wembley Stadium, Soldier Field, Old Trafford, Camp Nou
TerroristGroup (74)	Hezbollah, Khmer Rouge, Irish Republican Army, Shining Path, Tupac Amaru, Sendero Luminoso
Treaty (202)	North Atlantic Treaty, Kyoto Protocol, Louisiana Purchase, Montreal Protocol, Berne Convention
University (501)	University of Oslo, Stanford, CMU, Columbia University, Tsing Hua University, Cornell University
VideoGame (450)	Half Life, Final Fantasy, Grand Theft Auto, Warcraft, Need for Speed, Metal Gear, Gran Turismo
Wine (60)	Port, Champagne, Bordeaux, Rioja, Chardonnay, Merlot, Chianti, Cabernet Sauvignon, Pinot Noir
WorldWarBattle (127)	D-Day, Battle of Britain, Battle of the Bulge, Battle of Midway, Battle of the Somme, Battle of Crete

Table 1: Target classes with sizes of instance sets and examples of instances

most frequently in text processing and described in [6]: Cosine (the ubiquitous dot product); Jaccard (Jaccard's coefficient); Jensen-Shannon (the Jensen-Shannon divergence); L1-Norm; and Skew-Divergence.

Evaluation Procedure: Multiple lists of attributes are evaluated for each class, corresponding to the combination of the use of a particular vector similarity function with a particular choice of number of input instances per class, input seed attributes per class, and other system settings. To remove any undesirable psychological bias towards higher-ranked attributes during the assessment, the elements of each list to be evaluated are sorted alphabetically into a merged list. Each attribute of the merged list is manually assigned a correctness label within its respective class. Similarly to methodology previously proposed to evaluate answers to Definition questions [21], an attribute is *vital* if it must be present in an ideal list of attributes of the target class; *okay* if it provides useful but non-essential information; and *wrong* if it is incorrect. Thus, a correctness label is manually assigned to a total of 18,608 attributes extracted for the 40 target classes, in a process that once again confirms that evaluation of information extraction methods can be quite time consuming.

Label	Value	Examples of Attributes
vital	1.0	ProgLanguage: portability, Wine: taste
okay	0.5	Company: vision, NationalPark: reptiles
wrong	0.0	BasicFood: low carb, CarModel: driver

Table 2: Labels for assessing attribute correctness

To compute the overall precision score over a ranked list of extracted attributes, the correctness labels are converted to numeric values as shown in Table 2. Precision at some rank N in the list is thus measured as the sum of the assigned values of the first N candidate attributes, divided by N.

3.2 Quality of the Extracted Attributes

Figure 4 plots precision values for ranks 1 through 50, for each of the five similarity functions (Cosine vs. Jaccard vs. Jensen-Shannon vs. L1-Norm vs. Skew-Divergence). The first three graphs in Figure 4 show the precision over three individual target classes. Several conclusions can be drawn after inspecting the results. First, the quality of the attributes extracted by a given similarity function varies among classes. For instance, when using Jensen-Shannon

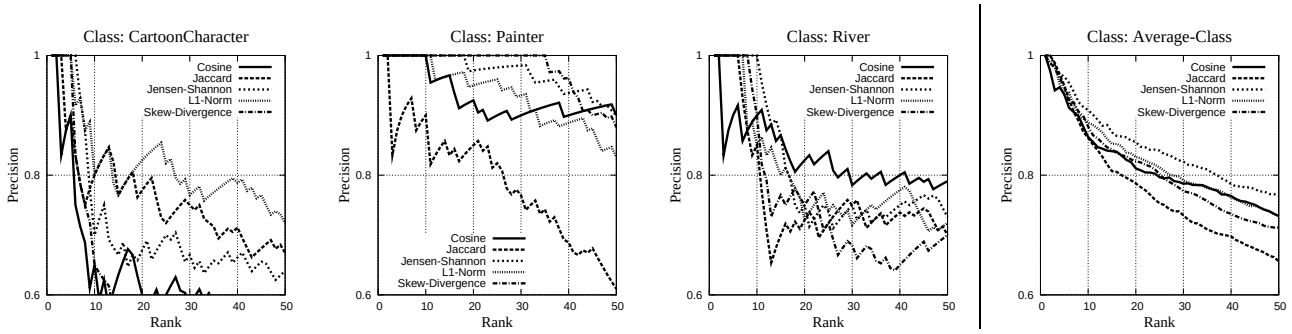

Figure 4: Influence of similarity functions on the precision of the ranked list of extracted attributes, for a few target classes (left) and as an average over all target classes (right)

Class	Top Extracted Attributes
Actor	awards, height, age, date of birth, weight, b** **** (offensive), birthdate, birthplace, cause of death, real name
AircraftModel	weight, length, history, fuel consumption, interior photos, specifications, photographs, interior pictures, seating arrangement, flight deck
Award	recipients, date, winners list, result, gossip, printable ballot, nominees, winners, location, announcements
BasicFood	calories, color, size, allergies, taste, carbs, nutritional information, nutrition facts, nutritional value, nutrition
CarModel	transmission, top speed, acceleration, transmission problems, owners manual, gas mileage, towing capacity, stalling, maintenance schedule, performance parts
CartoonChar	costume, voice, creator, first appearance, funny pictures, origins, cartoon images, cartoon pics, color pages,
CellPhoneModel	features, battery life, retail price, mobile review, specification, price list, functions, ratings, tips, tricks
ChemicalElem	mass, symbol, atomic number, normal phase, classification, electron configuration, atomic structure, origin name, freezing point, lewis dot diagram
City	map, population, zip code, climate, demographics, yellow pages, telephone directory, mayor, phone book,
Company	location, ceo, headquarters, stock price, mission statement, chairman, corporate office, company profile, code of conduct, 2004 annual report
Country	population, flag, climate, president, area, population density, geography, economy, religion, topography
Currency	exchange rates, symbol, currency converter, currency conversion, country, currency exchange, convert, exchange rate for, conversion chart, currency calculator
DigitalCamera	zoom, battery life, cost, troubleshooting, resolution, specification, specs, software download, uk price, instruction manual
Disease	treatment, symptoms, causes, incidence, signs, pictures, definition, cure, diagnosis, prognosis
Drug	side effects, dosage, price, withdrawal symptoms, generic equivalent, food source, dangers, mechanism of action, contraindications, half-life
Empire	collapse, size, definition, downfall, chronology, time line, location, achievements, accomplishments, rise
Flower	color, structure, genus, botanical name, anatomy, flower meaning, line drawing, taxonomy, sketch, oil paintings
Holiday	date, origin, customs, significance, history, definition, christian festival, meaning, purpose, traditions
Hurricane	damage, death toll, date, destruction, wind speed, satellite images, statistics, track, aftermath, prediction
Mountain	height, location, geology, elevation, topographic map, eruption history, lava type, formation, last eruption,
Movie	cast, release date, director, crew, cast list, synopsis, official site, official website, plot summary, storyline
NationalPark	location, geology, pics, address, climate, history, photographs, birds, photos, elevation
NbaTeam	owner, coach, video clips, jason williams, wallpapers, facts, message board, seating chart, mascot, championships
Newspaper	editor, website, owner, back issues, homepage, logo, publisher, tv guide, classified ads, circulation figures
Painter	paintings, biography, bibliography, autobiography, artwork, self portraits, quotations, bio, quotes, life history
ProgLanguage	syntax, advantages, basics, commands, tutorial, statement, examples, inventor, reference, help
Religion	beliefs, origin, gods, teachings, tenets, sacred writings, principles, practices, sacred texts, basics
River	location, length, mouth, tributaries, width, source, physical features, headwaters, depth, origin
SearchEngine	market share, share price, phone book, net worth, submit url, mission statement, owner, submissions, inventor,
SkyBody	distance, size, age, volume, diameter, radius, mass, surface gravity, orbital velocity, period of revolution
Skyscraper	height, architect, location, floors, dimensions, address, history, pics, floor plans, architecture
SoccerClub	league, capacity, chairman, titles, official site, official website, managers, tours, seating plan, past players
SportEvent	winners, events, champions, results, champs, dates, matchups, official site, official website, locations
Stadium	location, seating capacity, architect, address, seating map, dimensions, tours, pics, poster, box office
TerroristGroup	attacks, leader, goals, meaning, website, leadership, photos, images, definition, flag
Treaty	countries, ratification, date, definition, summary, purpose, pros, cons, members, picture
University	alumni, mascot, dean, economics department, career center, graduation 2005, department of psychology, school colors, tuition costs, campus map
VideoGame	price, system requirements, creator, official site, official website, free game download, concept art, download demo, pc cheat codes, reviews
Wine	vintage, color, cost, style, taste, vintage chart, pronunciation, shelf life, wine ratings, wine reviews
WorldWarBattle	date, location, significance, images, importance, timeline, summary, pics, maps, photographs

Table 3: Top attributes extracted using Jensen-Shannon as similarity function

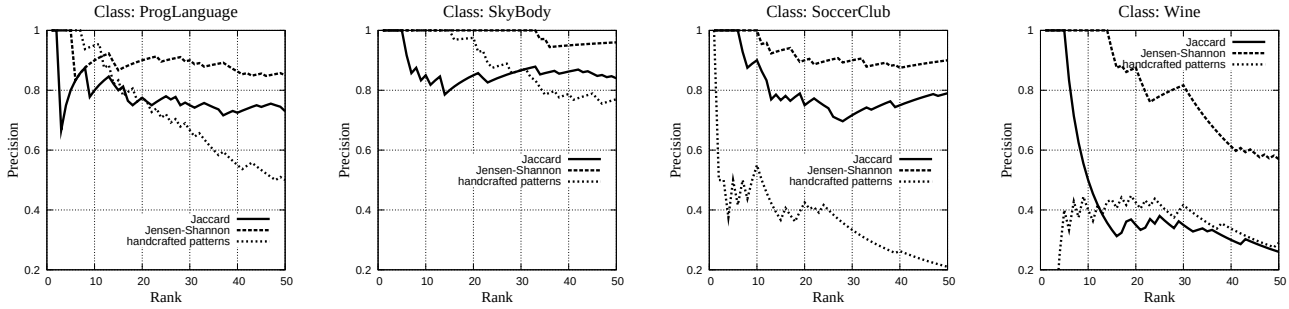

Figure 5: Relative performance of pattern-based extraction based on handcrafted patterns as proposed in previous work, vs. seed-based extraction proposed in this paper using Jaccard and Jensen-Shannon as similarity functions, for a few target classes

as similarity function, the attributes extracted for the class *Painter* are better than for *River*, which in turn are better than for *CartoonChar*. Second, not all similarity functions produce the same level of accuracy. The rightmost graph in Figure 4 shows the precision as an average over all target classes. Although none of the similarity functions outperforms the others on each and every target class, it turns out that, on average, Jensen-Shannon performs the best and Jaccard the worst, with L1-Norm, Cosine and Skew-Divergence placed in-between. For a more detailed view into the extracted attributes, Table 3 shows the top attributes extracted with Jensen-Shannon for each of the 40 target classes. Third, regardless of which of the five similarity functions is used, the precision as an average over all target classes is higher than 0.8 for rank 10, higher than 0.7 for rank 30, and higher than 0.6 for rank 50, as shown by the rightmost graph in Figure 4. These numbers are very high both in absolute value, but also relatively to the quality of attributes extracted with handcrafted patterns from query logs based on a previously-proposed method [13], as explained in the following.

3.3 Comparison to Previous Results

A recent study in information extraction from the Web [13] describes a method similar to this paper in goals (extraction of class attributes) and textual data source (extraction from query logs). The main difference is that, in that study, the extraction is fully supervised (rather than weakly supervised), using handcrafted extraction patterns (rather than seed attributes) to extract candidate attributes from queries that match those patterns. In the following, we denote the older, handcrafted-pattern method from [13] by P_{old}, and the seed-based method introduced in this paper by S_{new}. For a direct comparison of P_{old} and S_{new}, and since the evaluation of P_{old} in [13] reports results on only 5 target classes, the older method P_{old} was re-run in the experimental setting defined in this paper, with the same 40 target classes and the same set of search queries as data source. Figure 5 shows comparative precision curves for the classes *ProgLanguage*, *SkyBody*, *SoccerClub* and *Wine*. To avoid clutter, the performance of S_{new} is shown only for the two similarity functions that were shown earlier in Figure 4 to perform the best (Jensen-Shannon) and the worst (Jaccard).

The graphs in Figure 5 indicate that P_{old} has lower performance than S_{new} for the classes *SoccerClub* and *Wine*, and lower precision than S_{new} using Jensen-Shannon for the class *SkyBody*. In the case of the class *ProgLanguage*, P_{old} has higher accuracy than S_{new} for ranks 4 through 12, after which precision degrades more quickly as the rank in the list increases. To precisely quantify the differences between P_{old} and S_{new} using Jensen-Shannon, Table 4 provides an in-depth comparison of precision at ranks 10, 20 and 50, for each of the 40 target classes and as an average over all target classes. The seed-based approach S_{new} proposed in this paper clearly outperforms the previous method P_{old} from [13], with relative precision boosts of 25% (0.90 vs. 0.72) at rank 10, 32% (0.85 vs. 0.64) at rank 20, and 43% (0.76 vs. 0.53) at rank 50.

3.4 Minimizing the Amount of Supervision

The extraction of attributes from query logs requires a relatively small amount of supervision, which in our experiments is provided in the form of 5 seed attributes and a median of 172 instances per target class. While we believe that such a requirement is not at all unreasonable or impractical, quantifying the exact impact of reducing the amount of supervision on the quality of extracted attributes is still useful for at least two reasons. First, it offers an insight into the robustness of the method, and on its ability to perform the task at hand even in non-optimal conditions (that is, with scarce input data). More importantly, it gives a trustworthy estimate on the expected accuracy level in what could be called a fast-track development scenario, when the attributes for a new search vertical (with new target classes) need to be collected quickly with minimum effort and minimum input data.

Figure 6 illustrates the impact of providing less input data on the output quality. The graphs show the precision plots corresponding to reducing the number of instances, from the (regular) all down to 20 and then to 10 per class (left graph); and to reducing the number of seed attributes from the regular 5 down to 4, 3 and then 2 per class (right graph). As expected, the precision gradually decreases in both cases, but the decrease is small especially at lower ranks (1 through 20). In other words, the extraction method proposed in this paper can extract attributes of high quality even if it is given as few as 10 instances and 2 seed attributes per target class.

In a separate experiment, Steps 10 through 15 from the earlier Figure 1 are temporarily tweaked to generate a reference search-signature vector for only one of the target classes (randomly chosen to be *Country*), instead of generating one such reference vector for each class. This change further re-

Class	Precision						Class	Precision					
	@10		@20		@50			@10		@20		@50	
	P_{old}	S_{new}	P_{old}	S_{new}	P_{old}	S_{new}		P_{old}	S_{new}	P_{old}	S_{new}	P_{old}	S_{new}
Actor	0.85	1.00	0.82	1.00	0.74	0.96	Movie	0.95	1.00	0.90	0.95	0.72	0.85
AircraftModel	0.80	0.80	0.77	0.85	0.68	0.71	NationalPark	0.70	0.85	0.80	0.85	0.66	0.88
Award	0.30	0.95	0.15	0.77	0.24	0.69	NbaTeam	0.60	0.80	0.40	0.77	0.33	0.78
BasicFood	1.00	1.00	0.90	0.95	0.65	0.86	Newspaper	0.85	0.90	0.62	0.80	0.39	0.72
CarModel	0.95	1.00	0.77	1.00	0.77	0.89	Painter	1.00	1.00	0.95	0.97	0.86	0.90
CartoonChar	0.45	0.70	0.47	0.67	0.39	0.64	ProgLanguage	0.95	0.90	0.77	0.90	0.50	0.85
CellPhoneModel	0.55	0.90	0.57	0.87	0.23	0.78	Religion	1.00	1.00	0.92	0.95	0.78	0.95
ChemicalElem	0.90	0.80	0.67	0.80	0.71	0.83	River	0.75	1.00	0.70	0.75	0.55	0.73
City	0.20	0.75	0.20	0.75	0.31	0.68	SearchEngine	0.50	0.65	0.50	0.55	0.48	0.39
Company	0.90	1.00	0.82	0.97	0.79	0.85	SkyBody	1.00	1.00	0.97	1.00	0.77	0.96
Country	0.85	1.00	0.82	0.97	0.88	0.95	Skyscraper	0.85	0.95	0.60	0.87	0.48	0.74
Currency	0.50	0.80	0.25	0.67	0.16	0.36	SoccerClub	0.55	1.00	0.42	0.90	0.21	0.90
DigitalCamera	0.50	0.90	0.25	0.82	0.10	0.87	SportEvent	0.60	1.00	0.42	0.95	0.42	0.84
Disease	1.00	0.95	1.00	0.92	0.77	0.87	Stadium	0.75	0.90	0.72	0.85	0.57	0.83
Drug	1.00	0.90	0.87	0.90	0.84	0.81	TerroristGroup	0.55	0.90	0.62	0.82	0.43	0.49
Empire	0.85	0.90	0.77	0.87	0.66	0.82	Treaty	0.20	0.95	0.40	0.87	0.46	0.64
Flower	0.90	0.75	0.70	0.65	0.59	0.58	University	0.90	0.85	0.82	0.85	0.65	0.74
Holiday	0.65	0.90	0.60	0.65	0.36	0.52	VideoGame	0.70	0.90	0.57	0.90	0.44	0.90
Hurricane	1.00	0.95	0.87	0.95	0.76	0.73	Wine	0.40	1.00	0.42	0.87	0.29	0.57
Mountain	0.90	1.00	0.82	0.92	0.62	0.88	WorldWarBattle	0.00	0.85	0.00	0.82	0.00	0.66
							Average-Class	**0.72**	**0.90**	**0.64**	**0.85**	**0.53**	**0.76**

Table 4: Detailed relative performance of pattern-based extraction based on handcrafted patterns (P_{old}) as proposed in previous work, vs. seed-based extraction proposed in this paper (S_{new}) using Jensen-Shannon as similarity function

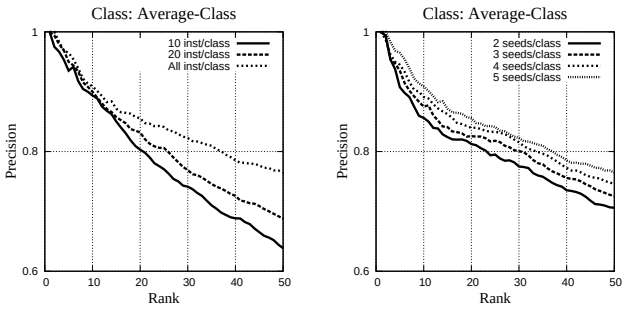

Figure 6: Impact of separately varying the number of input instances per class (left) or the number of seed attributes per class (right), on the precision of the ranked list of attributes extracted using Jensen-Shannon as similarity function, as an average over all target classes

duces the required amount of supervision, from 200 (5 times 40) seed attributes for all classes in the standard method, to 5 seed attributes for all classes in the tweaked experiment. The resulting precision values, as an average over all target classes, are 0.75 (at rank 10), 0.69 (at rank 20) and 0.56 (at rank 50). A quick comparison of these numbers with the last row of Table 4 shows that the precision is lower than with the standard configuration (S_{new} in Table 4), but still higher than the performance of the older, handcrafted-pattern method from [13] (P_{old} in Table 4).

3.5 Identifying Similar Attributes

The cumulative precision of each individual attribute, in a list of attributes extracted for a given class, is correlated with, but not a definitive measure of, the overall quality of the list. Attributes with a significant semantic overlap, such as the pairs *features* and *functions* for *CellPhoneModel*, or *photographs* and *photos* for *NationalPark*, or *tenets* and *principles* for *Religion* in Table 3, are relevant separately but less useful together, as they provide (near-)duplicate information. Even if a list of attributes has high precision, its diversity improves if attributes that are strongly related to each other are identified and grouped into equivalence (or semantic relatedness) classes.

To estimate the semantic relatedness between two given phrases, most approaches rely on external lexical resources created manually by experts, which organize concepts into rich thesauri (e.g., Roget's Thesaurus) or hierarchically (e.g., WordNet). The latter is the de-facto standard in computing semantic relatedness [1], although resources compiled collaboratively by non-experts (e.g., Wikipedia) represent intriguing alternatives [19]. Since WordNet is not designed to accommodate noisy, arbitrarily specific phrases (in this case, attributes) that occur in search queries, and in general to avoid using any manually compiled resources, it is preferable to identify similar attributes by exploiting lexical resources acquired from text by unsupervised methods. One such resource consists of pairs of phrases associated with distributional similarity scores, which measure the extent to which the component phrases occur in similar contexts in documents [9]. The scores have values in the interval [0,1]. For example, *tenets* and *principles* have a relatively high distributional similarity score, namely 0.39. The pairs and their scores are collected offline from 50 million news articles maintained by the Google search engine.

In an experiment designed as an initial exploration of whether distributional similarities could be used to identify similar attributes within a class, any two extracted attributes are automatically deemed to be similar, if their distributional similarity score is higher than 0.2 and each attribute is among the top 10 phrases that are most similar to the other attribute. As expected, distributional similar-

Manual Judgment	Examples of Attribute Pairs	
	Class	Attributes
Potentially useful:		
Synonyms	CartoonChar	quotations, sayings
	Country	gdp, gross domestic product
	Empire	administration, government
	Hurricane	path, route
	NationalPark	climate, weather
	Religion	gods, deities
	SoccerClub	emblem, logo
	WorldWarBattle	significance, importance
Strongly Related	AircraftModel	details, information
	Company	ceo, chairman
	Religion	gods, goddesses
	Stadium	layout, design
	Stadium	turf, grass
Hypernyms	BasicFood	nutrients, vitamins
	BasicFood	nutrients, antioxidants
	Painter	artwork, paintings
	Stadium	events, concerts
	TerroristGroup	attacks, bombings
Probably useless:		
Siblings	BasicFood	calories, carbs
	Mountain	longitude, latitude
	NationalPark	birds, animals
	Painter	paintings, drawings
	River	length, width
Incorrect	Actor	ethnicity, nationality
	Disease	symptoms, complications
	Disease	pathophysiology, etiology
	NationalPark	camping, hiking
	SportEvent	winners, finalists
	Stadium	renovation, construction

Table 5: Manual correctness judgments for various pairs of Top-50 extracted attributes found to be strongly related to one another based on distributional similarities

ities are a useful but limited criterion for finding similar attributes. As shown in the lower part of Table 5, some of the pairs of attributes automatically deemed to be similar are in fact useless when manually judging their correctness, either because the attributes are siblings (e.g., *length* and *width* for *River*) or simply because they are not equivalent (e.g., *symptoms* and *complications* for *Disease*). In contrast, the upper part of Table 5 shows pairs of attributes whose automatic identification as similar is potentially useful, including hypernyms (e.g., *attacks* is a hypernnym of *bombings* for *TerroristGroup*), strongly related phrases (e.g., *gods* and *goddesses* for *Religion*) and, ideally, synonyms (e.g., *climate* and *weather* for *NationalPark*). The manual judgment of all pairs of attributes found to be similar based on distributional similarities, across the top 50 attributes extracted for each of the 40 target classes, indicates that 50% of the pairs contain actual synonyms. Moreover, 79% of the pairs are potentially useful, as they contain synonyms, hypernyms or strongly related attributes. The results suggest that although distributional similarities alone are insufficient for finding similar attributes, they constitute an attractive, data-driven alternative to using manually constructed resources such as WordNet. In fact, distributional similarities identify pairs of attributes such as *climate* and *weather* for *NationalPark*, as well as *emblem* and *logo* for *SoccerClub*, to be similar although the respective pairs are not listed as synonyms in WordNet.

4. RELATED WORK

In contrast to previous approaches to large-scale information extraction, which rely exclusively on large document collections, for mining pre-specified relations, we explore the role of query logs in extracting unrestricted types of relations, namely class attributes. A related recent approach [18] pursues the goal of unrestricted relation discovery from textual documents.

Our extracted attributes are relations among objects in the given class, and objects or values from other, "hidden" classes. Determining the type of the "hidden" argument of each attribute (e.g., *Person* and *Location* for the attributes *chief executive officer* and *headquarters* of the class *Company*) is beyond the scope of this paper. Nevertheless, the lists of extracted attributes have direct benefits in gauging existing methods for harvesting pre-specified semantic relations [2, 14], 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 drugs.

In [3], the acquisition of attributes and other knowledge relies on Web users who explicitly specify it by hand. In contrast, we may think of our approach as Web users implicitly giving us the same type of information, outside of any systematic attempts to collect knowledge of general use from the users. The method proposed in [20] applies handcrafted lexico-syntactic patterns to text within a small collection of Web documents. The resulting attributes are evaluated through a notion of question answerability, wherein an attribute is judged to be valid if a question can be formulated about it. More precisely, evaluation consists of users manually assessing how natural the resulting candidate attributes are, when placed in a *wh-* question. Comparatively, our evaluation is stricter. Indeed, many attributes, such as *long term uses* and *users* for the class *Drug*, are marked as wrong in our evaluation, although they would easily pass the question answerability test (e.g., *"What are the long term uses of Prilosec?"*) used in [20]. Because our evaluation is stricter, a direct comparison of our precision numbers with those reported in [20] is not possible. However, the lists of top attributes shown in Table 3 for *City* and *River* can be compared against their equivalent lists reported in [20] for the classes *Town* and *River*, shown below for reference:

• *Town*: [population, history, home page, sightseeing, info, finance, facility, heritage, environment, hot spring];

• *River*: [water level, upstream, name, environment, water quality, history, head stream, picture, water, surface] [20].

Although query logs received much attention in the task of improving information retrieval, they were explored as a resource for acquiring explicit relations in information extraction only recently [13]. Comparatively, the attribute extraction method introduced here replaces handcrafted patterns with seed attributes, pursues a larger-scale evaluation over 40 instead of only 5 target classes, and operates with significantly higher accuracy as described in Section 3.

5. CONCLUSION

Traditional wisdom suggests that textual documents tend to assert information (statements or facts) about the world in the form of expository text. Comparatively, search queries can be thought of as being nothing more than noisy, keyword-based approximations of often-underspecified user information needs (interrogations). Despite this apparent disad-

vantage, and in a departure from previous approaches to large-scale information extraction from the Web, this paper introduces a weakly-supervised extraction framework for mining useful knowledge from query logs, rather than Web documents. The framework lends itself to a concrete Web mining task, namely class attribute extraction. In an evaluation of attributes extracted for a variety of domains of interest to Web search users, the quality of the resulting attributes exceeds previously reported results by 25% at rank 10, and 43% at rank 50, thus holding the promise of a new path in research in information extraction from query logs.

Since the extracted attributes correspond to types of facts collected from actual search queries submitted by Web users, they constitute a building block towards acquiring large repositories of facts from Web documents, and exploiting them during Web search. Ongoing work spans diversification towards other languages; the development of open-domain methods for populating attributes corresponding to non-traditional types of facts (e.g., *side effects* for *Drug* and *seating arrangement* for *AircraftModel*, rather than traditionally studied cases like *population* for *Country* or *height* for *Mountain*); an exploration of the role of query logs in other information extraction tasks; and the integration of signals from documents in addition to query logs.

6. ACKNOWLEDGMENTS

The author would like to thank Hang Cui, for comments on an early draft; Nikesh Garera and Benjamin Van Durme, for implementing the computation of pairwise similarities of search-signature vectors, writing scripts for determining precision scores and manually evaluating a subset of the candidate attributes extracted for 5 target classes; and Dekang Lin, for collecting and providing access to distributionally similar phrases.

7. REFERENCES

[1] A. Budanitsky and G. Hirst. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 2006.

[2] 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.

[3] 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.

[4] H. Cui, J. Wen, J. Nie, and W. Ma. Probabilistic query qxpansion using query logs. In *Proceedings of the 11th World Wide Web Conference (WWW-02)*, pages 325–332, Honolulu, Hawaii, 2002.

[5] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: Is more always better? In *Proceedings of the 24th ACM Conference on Research and Development in Information Retrieval (SIGIR-02)*, pages 207–214, Tampere, Finland, 2002.

[6] L. Lee. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association of Computational Linguistics (ACL-99)*, pages 25–32, College Park, Maryland, 1999.

[7] M. Li, M. Zhu, Y. Zhang, and M. Zhou. Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 1025–1032, Sydney, Australia, 2006.

[8] X. Li and D. Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-02)*, pages 556–562, Taipei, Taiwan, 2002.

[9] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-98)*, pages 768–774, Montreal, Quebec, 1998.

[10] L. Lita and J. Carbonell. Instance-based question answering: A data driven approach. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pages 396–403, Barcelona, Spain, 2004.

[11] R. Mooney and R. Bunescu. Mining knowledge from text using information extraction. *SIGKDD Explorations*, 7(1):3–10, 2005.

[12] M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Organizing and searching the World Wide Web of facts - step one: the one-million fact extraction challenge. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pages 1400–1405, Boston, Massachusetts, 2006.

[13] M. Paşca and B. Van Durme. What you seek is what you get: Extraction of class attributes from query logs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2832–2837, Hyderabad, India, 2007.

[14] 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.

[15] P. Pantel and D. Ravichandran. Automatically labeling semantic classes. In *Proceedings of the 2004 Human Language Technology Conference (HLT-NAACL-04)*, pages 321–328, Boston, Massachusetts, 2004.

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

[17] L. Schubert. Turing's dream and the knowledge challenge. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, Boston, Massachusetts, 2006.

[18] Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the 2006 Human Language Technology Conference (HLT-NAACL-06)*, pages 204–311, New York, New York, 2006.

[19] M. Strube and S. Ponzetto. Wikirelate! computing semantic relatedness using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pages 1419–1424, Boston, Massachusetts, 2006.

[20] 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.

[21] E. Voorhees. Evaluating answers to definition questions. In *Proceedings of the 2003 Human Language Technology Conference (HLT-NAACL-03)*, pages 109–111, Edmonton, Canada, 2003.

[22] Z. Zhuang and S. Cucerzan. Re-ranking search results using query logs. In *Proceedings of the 15th International Conference on Information and Knowledge Management (CIKM-06)*, Arlington, Virginia, 2006.