Attribute Extraction from Synthetic Web Search Queries

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

Abstract

The accuracy and coverage of existing methods for extracting attributes of instances from text in general, and Web search queries in particular, are limited by two main factors: availability of input textual data to which the methods can be applied, and inherent limitations of the underlying assumptions and algorithms being used. This paper proposes a weakly-supervised approach for the acquisition of attributes of instances from input data available in the form of synthetic queries automatically generated from submitted queries. The generated queries allow for the acquisition of additional attributes, leading to extracted lists of attributes of higher quality than with comparable previous methods.

1 Introduction

Motivation: The availability of larger textual data sources has allowed research in information extraction to shift its focus towards robust methods that require little or no annotated data, operate at large scale with lower computational costs, and acquire open-domain information (Banko and Etzioni, 2008). The information is usually targeted at three levels of granularity: classes (e.g., *giant planets*), class elements or instances (e.g., *jupiter*, *uranus*, *saturn*), and relations among instances. Since these types of information would form the backbone of knowledge bases acquired automatically from text (Mooney and Bunescu, 2005), their acquisition has received increased attention over recent years.

Among other types of relations targeted by various extraction methods, attributes (e.g., *escape velocity*, *diameter* and *surface gravity*) have emerged as one of the more popular types, as they capture quantifiable properties of their respective classes (*giant planets*) and instances (*jupiter*). A variety of attribute extraction methods mine textual data sources ranging from unstructured (Tokunaga et al., 2005) or structured (Cafarella et al., 2008) text within Web documents, to human-compiled encyclopedia (Wu et al., 2008; Cui et al., 2009) and Web search query logs (Alfonseca et al., 2010), attempting to extract, for a given class or instance, a ranked list of attributes that is as comprehensive and accurate as possible. The accuracy and coverage of existing methods (Raju et al., 2008; Alfonseca et al., 2010) for extracting attributes of instances are limited by two main factors: availability of input textual data to which the methods can be applied; and inherent limitations of the underlying assumptions and algorithms being used. For example, a simple but effective method was proposed in (Paşca and Van Durme, 2007) for extracting attributes of an instance, by applying a small set of extraction patterns (e.g., *A of I*) to Web search queries (e.g., *"escape velocity of jupiter"*). If the input set of queries increased, additional candidate attributes would be extracted.

Contributions: This paper introduces a weakly-supervised approach for the acquisition of attributes of instances from query logs, by automatically expanding the set of known (organic) queries from which attributes are extracted with additional, inferred (synthetic), not-yet-submitted queries. The focus on expanding the input textual data gives a generally-applicable approach, which can be applied to existing methods for attribute acquisition from query logs, to increase coverage. In particular, the application of previously-proposed extraction patterns (Paşca and Van Durme, 2007)

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 401–409,
Chiang Mai, Thailand, November 8 – 13, 2011.

to the expanded set of queries allows for the acquisition of additional attributes that would otherwise not be acquired only from the set of known queries.

In order to infer new queries, known queries are aggregated into query templates (e.g., *"lyrics of ⋆ beatles"*) associated with known phrase fillers (e.g., ⋆→{*yesterday*, *hey jude*}). The known phrase fillers of each query template are then expanded into new candidate phrase fillers. In contrast to previous work on query generation (Mitkov and Ha, 2006; Heilman and Smith, 2010), new queries are generated based on query analysis alone, as opposed to individual document analysis. This has the potential advantages of scalability and robustness when applied to arbitrary, inherently-noisy queries. Among the inferred queries, the ones of higher interest to attribute extraction are those derived from a query template that fixes either a potential attribute (e.g., *"surface gravity of ⋆"*) or a potential instance (e.g., *"⋆ of jupiter"*). In experiments using a large set of anonymized search queries, the inferred queries allow for the acquisition of accurate attributes over an evaluation set of 75 instances introduced in previous work (Alfonseca et al., 2010).
Applications: Attributes are useful in information retrieval, e.g., for suggesting related queries (Bellare et al., 2007) and recommending products (Probst et al., 2007). They plan an important role in knowledge acquisition and representation (Guarino, 1992), for example as building blocks in the manual compilation of infoboxes in Wikipedia (Remy, 2002). Furthermore, the availability of a larger number of more accurate attributes allows for the development of better search interfaces geared towards structured search. Examples of such interfaces are Wolfram Alpha and Google Squared, two search tools that can take as input instances and return lists of attributes and their values.

2 Attribute Extraction

2.1 Intuitions and Scope

Intuitions: Our attribute extraction method is inspired by several intuitions. First, phrases of the same class (car models, search engines, baseball players etc.) share similar properties or attributes, enter similar relations and satisfy similar constraints. For example, car models have replacement parts, are assembled by some maker in some year, and have an estimated current value. Consequently, known queries that refer to similar phrases may overlap significantly, or even become equal once the phrases have been replaced by a common slot filled by the phrases. Thus, *"kelley blue book value of 2008 dodge charger"* and *"kelley blue book value of 2008 honda civic"* can be grouped into a shared query template *"kelley blue book of 2008 ⋆"*, whose slot ⋆ is filled by the names of car models. Second, known queries that can be grouped into a shared query template often provide a small sample of, rather than comprehensive coverage of, all phrases that would meaningfully fill the template. Therefore, new queries can be generated by filling the slots of query templates with new phrase fillers similar to known fillers. For instance, *"kelley blue book value of 2008 chrysler sebring"* can be inferred as a new candidate query if *chrysler sebring* is known to be similar to *dodge charger* and/or *honda civic*. Third, a new candidate query is meaningful if the new phrase filler is similar to the known fillers not only statically, but also in the context of the query template. This is particularly important for query templates with few, ambiguous phrase fillers. Consider the query template *"lyrics of ⋆ beatles"*, with the known fillers *come together*, *hey jude* and *yesterday*. Although phrases such as *gather together*, *earlier today* and *last friday* are highly similar to the known fillers, they are not meaningful new fillers for *"lyrics of ⋆ beatles"* and would therefore produce spurious new queries. In contrast, *lovely rita* and *here comes the sun* are similar to the known fillers both statically and in the context of the query template.

Scope: Following the above intuitions, attributes can be extracted from generated queries. In turn, generated queries are inferred by essentially replacing phrases from known queries with meaningful, similar phrases. The form (e.g., long and complex, vs. short and simple) and scope (e.g., open-domain vs. domain-specific) of known queries determine the form and scope of inferred queries. While this prevents arbitrarily complex queries from being generated, it has the advantage of homogeneity of new queries relative to known queries. Also, this should have little, if any, impact on extracted attributes, since the latter are actually extracted from queries whose form is relatively simple rather than complex. The scope of new queries is further influenced by the availabil-

ity of new phrases that are similar to phrases from known queries. For example, *chrysler sebring* must be available as a phrase similar to *dodge charger* and/or *honda civic*, in order to potentially generate *"kelley blue book of 2008 chrysler sebring"* from the known queries *"kelley blue book of 2008 dodge charger"* and *"kelley blue book of 2008 honda civic"*.

2.2 Extraction from Generated Queries

Aggregation into Query Templates: The input to the method is a set of Web search queries. As described in (Paşca, 2011), the sequence of terms available in each query is split into all combinations of triples of a prefix, non-empty infix and postfix. Queries that share a common prefix and common postfix are aggregated into a query template, where the input infixes are the known phrase fillers of the template. For example, queries such as *"lyrics of yesterday beatles"* and *"lyrics of come together beatles"* are aggregated into the template *"lyrics of* $\star$ *beatles"*, where the template filler $\star$ corresponds to the set of known phrase fillers, i.e., infixes from the input queries: {*yesterday*, *come together*}. An input query may contribute to the creation of multiple query templates, via different infixes. For example, another template created from *"lyrics of yesterday beatles" "lyrics of yesterday toni braxton"* is *"lyrics of yesterday* $\star$*"*.

Like the subsequent stages of processing, generating all possible infixes of all input queries, especially for large input sets of queries, is a non-trivial computational challenge. However, the computation can be translated into parallelizable operations in a distributed computing framework such as Hadoop (White, 2010) or MapReduce (Dean and Ghemawat, 2004). In particular, the aggregation of queries into templates can be performed in a single MapReduce step. The mapper takes as input queries, and splits them into one or more mappings from a query template (key) to a corresponding infix, i.e., a known phrase filler (value). For each query template, the reducer simply aggregates its phrase fillers into a set.

Generation of Candidate Phrase Fillers: In order to generate new queries, the set of known phrase fillers is expanded into additional candidate phrases that may fill the query template. As a prerequisite to generating candidate phrase fillers, distributionally similar phrases (Lin and Pantel, 2002; Lin and Wu, 2009; Pantel et al., 2009) and their scores are collected in advance. The assumption is that phrases that appear in similar contexts have similar meanings. A phrase is represented as a vector of its contextual features. A feature is a token, collected from windows of three tokens centered around the occurrences of the phrase in sentences across Web documents (Lin and Wu, 2009). Alternatively, the context could be approximated via linguistic dependencies detected with noun chunking (Pantel et al., 2009) and syntactic parsing (Lin and Pantel, 2002). In the contextual vector of a phrase, the weight of a feature is the pointwise-mutual information (Lin and Wu, 2009) between the phrase P and the feature F:

$$PMI(P,F) = \log \frac{Freq(P,F) \times N}{Freq(P) \times Freq(F)} \quad (1)$$

where $Freq(P,F)$ is the frequency of the feature F occurring with the phrase P, and N is the feature vocabulary size. The distributional similarity score between two phrases P_1 and P_2 is the cosine similarity between the contextual vectors of the two phrases. Alternatively, vector similarity could be computed via the Jaccard or Dice coefficients (Pantel et al., 2009). The lists $DS(P)$ of most distributionally similar phrases of a phrase P are thus compiled offline, by ranking the similar phrases of P in decreasing order of their $DSscore$ relative to P.

The most distributionally similar (Lin and Pantel, 2002; Pantel et al., 2009) phrases, of a known phrase filler K_i from a query template T, are considered to be candidate phrase filler U of the respective query template. The score of a candidate relative to the entire set of known fillers is the average of the distributional similarity scores between the candidate and each known filler. For each template T, its candidate phrase fillers U are ranked in decreasing order of their scores. Known phrase fillers of T are discarded from the resulting list of candidate phrase fillers of T.

The generation of candidate phrase fillers translates into two MapReduce steps. The first step rearranges the mappings from a query template to a set of known phrase fillers, into mappings from a known phrase filler to a set of query templates. Concretely, the mapper takes as input mappings from a query template (key) to its set of known phrase fillers (value), as they were output after the aggregation into query templates. The mapper emits mappings from a known phrase filler (key)

to a query template (value). For each phrase filler, the reducer aggregates its query templates into a set. The second MapReduce step takes this data, and joins it with distributional similarity data. The latter is available as mappings from a phrase (key) to a list of scored similar phrases (value). The mapper selects similar phrases as candidate phrase fillers for the template, as explained earlier. The output of the second step consists in mappings from a query template (key) to its set of known phrase fillers, as well as to a list of scored candidate phrase fillers (value).

Filtering of Candidate Phrase Fillers: Candidate phrase fillers generated via distributional similarities are similar to known phrases only statically. To also take the context of the query template into account, the candidates are filtered using the input queries. More precisely, the list of candidate phrases of a query template T is filtered, by retaining only known phrases of some other templates T' that are equivalent to T. Templates are deemed equivalent if they become identical after removal of stop words and other linking particles (prepositions, conjunctions etc.), term stemming and term reordering. For example, if the unfiltered candidate phrase *eleanor rigby* for the template *"lyrics of ⋆ beatles"* appears as a known phrase filler of the template *"lyrics for ⋆ by the beatles"*, then *eleanor rigby* is retained after filtering for the template *"lyrics of ⋆ beatles"*.

The filtering of candidate phrases using equivalent templates is modeled as two MapReduce steps. The first step takes as input mappings from a query template (key) to its set of known phrase fillers and list of scored candidate phrase fillers (value), as they were output after the generation of candidate phrase fillers. The mapper converts the query template into the corresponding equivalent template that contains the individual terms in lexicographic order. It emits mappings from an equivalent template (key), to a query template with its set of known phrase fillers and its list of scored candidate phrase fillers (value). For each equivalent template, the reducer simply aggregates this data. The second step takes as input mappings from an equivalent template (key) to its query templates with their sets of known phrase fillers and lists of scored candidate phrase fillers (value). For each equivalent template, the mapper iterates over its query templates. It checks which of its candidate phrase fillers occur among the known phrase fillers of the other query templates. The candidate phrase fillers that pass this test are retained as filtered phrase fillers. The mapper emits mappings from a query template (key) to its set of known phrase fillers, list of scored unfiltered phrase fillers, and list of scored filtered phrase fillers. The reducer merely emits its input, without modifications.

The relative ranking of candidate phrases from the list of inferred unfiltered phrase fillers (before filtering) is preserved in the list of inferred filtered phrase fillers (after filtering). Each filtered phrase filler inferred for a query template corresponds to a new query, generated by filling the phrase into the slot filler of the query template.

Attribute Extraction: Extraction patterns such as *"A of I"*, introduced in previous work (Paşca and Van Durme, 2007) to extract a candidate attribute A for a candidate instance I from queries, can be immediately applied to inferred queries. Thus, additional candidate attributes can be extracted from inferred queries, where the inferred queries are obtained via two types of query templates:

• query templates that specify a potential instance, and leave the attribute (e.g., A in *"A of I"*) as a phrase filler being inferred:

surface gravity of jupiter
europa of jupiter
composition of atmosphere of jupiter
father of jupiter

} "⋆ of jupiter"

Each inferred phrase filler (e.g., *core temperature*, *luminosity*) is collected as a candidate attribute of the phrase specified in the query template (*jupiter*). In this case, attribute extraction is equivalent to transferring an instance associated with a noisy set of attributes that are known phrase fillers of a template, to be associated with new attributes that are inferred phrase fillers of the template.

• query templates that specify a potential attribute, and leave the instance (e.g., I in *"A of I"*) as a phrase filler being inferred:

mass of positronium
mass of planets in order
mass of steel beams
mass of planet uranus

} "mass of ⋆"

For each inferred phrase filler (e.g., *cesium 137*, *water drop*), the phrase specified in the query template (*mass*) is collected as a candidate attribute. In this case, attribute extraction is equivalent to transferring an attribute associated with a noisy set

Phrase	Ranked List Available in Data Repository
caesium	[cesium, rubidium, strontium, barium, thallium, lanthanum, potassium, cerium, yttrium, bismuth, indium, gallium, europium, cadmium, antimony, ammonium,..]
ch3br	[ch3cl, ch4, nh3, ch 4, c2h4, ch3i, nh 3, ch3oh, c2h2, ch3f, c2h6, hcho, no2, h2s, hcn, ch30h, n2o, n20, ch3cn, hcooh, ethane, cc14, ethene, propene, hzo, c02, ch3sh, chbr3,..]
exxon valdez	[torrey canyon, amoco cadiz, sea empress, braer, 1989 exxon valdez, valdez oil, exxon valdez oil tanker, chernobyl nuclear, cosco busan, valdez oil spill, chernobyl,..]
fragile x	[fragile x, klinefelter syndrome, rett syndrome, fxtas, turner syndrome, down syndrome, friedreich ataxia, myotonic dystrophy, huntington's disease, williams syndrome, trisomy 21,..]
medal	[congressional medal of honor, navy cross, distinguished service cross, victoria cross, distinguished flying cross,air force cross, bronze star, purple heart, soldier's medal, medal honor, military cross,..]
men and women	[women and men, men and woman, males and females, men & women, young men, women, people, men, individuals, boys and girls, girls and boys, adults, sexes, females and males,..]
nerve	[ganglion, preganglionic, postganglionic, peripheral nerve, sciatic, sciatic nerve, axon, afferent, nerve root, dorsal root, ganglionic, vagus, trigeminal, median nerve,..]
yesterday	[last week, earlier today, last friday, last night, two days ago, yesturday, last thursday, earlier this week, just yesterday, last wednesday, yesteday, last monday, last month, two weeks ago,..]
whey	[protein powder, whey protein isolate, whey protein powder, soy protein, whey isolate, protein whey, casein protein, creatine, creatine monohydrate, isopure,..]

Table 1: Examples of ranked lists of similar phrases, available in the phrase similarity repository for various phrases

of instances that are known phrase fillers of a template, to be associated with new instances that are inferred phrase fillers of the template.

Regardless of whether they are specified manually or derived automatically, extraction patterns used in information extraction are imperfect (Kozareva et al., 2008). The pattern *A of I* for attribute extraction is no exception. The two types of query templates from above have known phrase fillers that are not true attributes (e.g., *europa* for the instance *jupiter*) and instances (e.g., *planets in order* for the attribute *mass*) respectively. This phenomenon is not a defect of this particular approach, but is inherited from and shared with any methods using such patterns for attribute extraction, as well as with any methods that rely on seed attributes, when the seeds are noisy rather than clean.

Attribute Ranking: As explained earlier, the score of an inferred phrase filler is computed as the average of similarity scores relative to known phrase fillers. The score is assigned to the pair of an attribute and instance extracted from the phrase filler. Attributes extracted for an instance are ranked in decreasing order of the scores.

3 Experimental Setting

Textual Data Sources: The acquisition of instance attributes relies on a random sample of around 100 million fully-anonymized queries in English submitted by Web users in 2010. Each query is accompanied by its frequency of occurrence in the query logs.

A phrase similarity repository is derived following (Pantel et al., 2009), from unstructured text available within a sample of around 200 million documents in English. The repository provides data for each of around 1 million phrases that occur as full-length queries in the input query logs. It contains ranked lists of the top 200 phrases computed to be the most distributionally similar, for each phrase. Table 1 illustrates the actual

aaa, ac compressors, acheron, acrocyanosis, adelaide cbd, african population, agua caliente casino, al hirschfeld, alessandro nesta, american fascism, american society for horticultural science, ancient babylonia, angioplasty, annapolis harbor, antarctic region, arlene martel, arrabiata sauce, artificial intelligence, bangla music, baquba, bb gun, berkshire hathaway, bicalutamide, blue jay, boulder colorado, brittle star, capsicum, carbonate, carotid arteries, chester arthur, christian songs, cloxacillin, cobol, communicable diseases, contemporary art, cortex, ct scan, digital fortress, eartha kitt, eating disorders, file sharing, final fantasy vii, forensics, habbo hotel, halogens, halophytes, ho chi minh trail, icici prudential, jane fonda, juan carlos, karlsruhe, kidney stones, lipoma, loss of appetite, lucky ali, majorca, martin frobisher, mexico city, pancho villa, phosphorus, playing cards, prednisone, right to vote, robotics, rouen, scientific revolution, self-esteem, spandex, strattera, u.s., vida guerra, visual basic, web hosting, windsurfing, wlan

Table 2: Set of 75 target instances, used in the evaluation of instance attribute extraction

ranked lists available in the repository for various phrases. The underlying similarity score between two phrases is the cosine between their vectors of context windows.

Target Instances: The performance of attribute extraction is computed over a standard set of 75 instances, previously introduced in (Alfonseca et al., 2010). As shown in Table 2, the set of instances ensures varied experimentation across multiple domains.

Experimental Runs: The experiments consist of several individual runs. Runs R_U and R_F acquire attributes from queries inferred via the first type of target query templates (e.g., *"⋆ of jupiter"*), before filtering (R_U) and after filtering (R_F). Run R_I uses queries inferred via the second type of target query templates (e.g., *"mass of ⋆"*), after filtering.

In order to compare with existing work, a previous extraction method from (Paşca and Van Durme, 2007), which uses extraction patterns, is implemented in a baseline run R_P. For consistency, the data source to the run R_P is the same set of input queries described at the beginning of the section.

Label	Value	Examples of Attributes
vital	1.0	capsicum: calorie count
		cloxacillin: side effects
		lucky ali: album songs
okay	0.5	jane fonda: musical theatre contributions
		mexico city: cathedral
		robotics: three laws
wrong	0.0	acheron: kingdom
		berkshire hathaway: tax exclusion
		contemporary art: urban institute

Table 3: Correctness labels manually assigned to attributes extracted for various instances

The per-instance ranked lists of attributes produced by the individual runs from above are concatenated in a series of combination runs. For example, run R_{FP} concatenates the attributes output by R_F and by R_P, in this order.

Evaluation Procedure: The evaluation focuses on the assessment of accuracy of the ranked list of attributes generated for each instance. To remove any undesirable bias towards higher-ranked attributes, the attributes 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 relative to its respective instance. In accordance with previously introduced methodology, an attribute is *vital* if it must be present in an ideal list of attributes of the instance (e.g., *side effects* for *cloxacillin*); *okay* if it provides useful but non-essential information; and *wrong* if it is incorrect (Paşca, 2007). Thus, a correctness label is manually assigned to a total of 4,833 attributes extracted for the 75 target instances.

To compute the precision score over a ranked list of attributes, the correctness labels are converted to numeric values (*vital* to 1, *okay* to 0.5 and *wrong* to 0), as shown in Table 3. Precision at some rank N in the list is measured as the sum of the correctness values of the attributes extracted up to rank N, divided by the number of those attributes.

4 Evaluation Results

Attribute Accuracy: Table 4 compares precision at various ranks, in the ranked lists of attributes

Run	Precision of Ranked Attributes				
	@1	@5	@10	@20	@50
Average (over all):					
R_P	0.61	**0.65**	**0.65**	**0.67**	**0.65**
R_I	0.61	0.62	0.60	0.61	0.60
R_U	0.45	0.37	0.33	0.31	0.30
R_F	0.64	0.58	0.57	0.55	0.51
R_{IP}	0.68	0.68	0.67	0.67	0.66
R_{IF}	0.64	0.64	0.63	0.64	0.62
R_{FP}	**0.77**	**0.69**	**0.68**	**0.65**	**0.62**
R_{FI}	**0.77**	**0.69**	**0.68**	**0.65**	**0.62**
Average (over non-empty):					
R_P	0.66	0.71	0.71	0.73	0.71
R_I	0.67	0.67	0.65	0.67	0.65
R_U	0.59	0.48	0.43	0.41	0.38
R_F	**0.83**	**0.75**	**0.74**	**0.71**	**0.66**
R_{IP}	0.69	0.69	0.67	0.68	**0.67**
R_{IF}	0.68	0.68	0.66	0.67	0.66
R_{FP}	**0.83**	**0.74**	**0.72**	**0.70**	**0.67**
R_{FI}	0.81	0.73	**0.72**	0.69	0.65

Table 4: Comparative accuracy of the ranked lists of attributes extracted in various runs, as an average over the entire set of 75 instances; and as an average over the (variable) subsets of instances for which some attributes were extracted

extracted by various runs. In the upper half of the table, average precision scores penalize instances for which no attributes are extracted. In contrast, the average scores in the lower half of the table only consider the instances for which some attributes are extracted.

The runs R_I, R_U and R_F operate directly over generated queries. One of the two types of query templates that produce attributes performs better, as illustrated by lower scores with R_F than with R_I. The difference in scores between R_U and R_F, which are about twice as high at rank 50 for the latter, illustrates the positive impact of filtering the candidate phrase fillers inferred from the query templates.

The benefit of combining the output from individual runs is illustrated by the generally higher scores given by combination runs in Table 4, relative to individual runs that they combine. Among combination runs, R_{FP} gives the highest scores at most ranks. When considering the accuracy of all runs, the highest scores over the entire evaluation set of instances are given by the combination run R_{FP}. Over subsets of instances with non-empty attribute lists, the accuracy of the individual run R_F is higher than all other individual runs, at par with the combination run R_{FP}. Table 5 shows the ranked lists of attributes extracted by R_F for a sample of instances.

Instance	[Ranked List of Inferred Attributes]
aaa	[symptoms, epidemiology, differential diagnosis, mortality, risk, surgical repair, stenting, resection, benefits, stent placement,..]
ac compressors	[manufacturer]
berkshire hathaway	[net asset value, closing price, total assets, dividend yield, par value, fair value, shares outstanding, class a shares, current price, common stock,..]
martin frobisher	[time line, routes, voyage, bibliography, explorations, ships, discoveries, travel,..]
scientific revolution	[negative effects, social impacts, social impact, theories, positive effects, accomplishments, long term effects, influences, cause and effects, scientific discoveries,..]
wlan	[limitations, risks, architectures, benefits, configurations, throughput, concepts, config, physical layer, security vulnerabilities,..]

Table 5: Examples of ranked lists of attributes extracted in run R_F from inferred filtered queries. None of these attributes are extracted for the respective instances in the baseline run R_P

5 Related Work

Previous work on attribute extraction uses a variety of types of textual data as sources for mining attributes. Taking advantage of structured and semi-structured text available within Web documents, the method introduced in (Yoshinaga and Torisawa, 2007) assembles and submits list-seeking queries to general-purpose Web search engines, and analyzes the retrieved documents to identify common structural (HTML) patterns

around class labels given as input, and potential attributes. Similarly, layout (e.g., font color and size) and other HTML tags serve as clues to acquire attributes from either domain-specific documents such as those from product and auction Web sites (Wong et al., 2008) or from arbitrary documents, optionally relying on the presence of explicit itemized lists or tables (Cafarella et al., 2008). As an alternative to Web documents, articles within online encyclopedia can also be exploited as sources of structured text for attribute extraction, as illustrated by previous work using infoboxes and category labels (Suchanek et al., 2007; Nastase and Strube, 2008; Wu and Weld, 2008) associated with articles within Wikipedia.

Working with unstructured text within Web documents, the method described in (Tokunaga et al., 2005) applies manually-created lexico-syntactic patterns to document sentences in order to extract candidate attributes, given various class labels as input. The candidate attributes are ranked using several frequency statistics. If the documents are domain-specific, such as documents containing product reviews, additional heuristically-motivated filters and scoring metrics can be used to extract and rank the attributes (Raju et al., 2008). In (Bellare et al., 2007), the extraction is guided by a small set of manually-provided seed instances and attributes rather than manually-created patterns, with the purpose of generating training data and extract new pairs of instances and attributes from text.

Web search queries have also been considered as a textual data source for attribute extraction, using extraction patterns (Paşca and Van Durme, 2007) or seed attributes (Paşca, 2007) to guide the extraction, and leading to attributes of higher accuracy than those extracted with equivalent techniques from Web documents. If the input data includes query sessions in addition to sets of search queries, extracted attributes have higher quality (Paşca et al., 2010). Given an instance (e.g., *nissan gt-r*) and a numerical attribute (e.g., *width*) extracted with a method like ours, the acquisition of the corresponding values (e.g., *1.9m*) is the aim of other research endeavors (Davidov and Rappoport, 2010; Bakalov et al., 2011).

6 Conclusion

The role of Web search queries in information extraction has been previously explored. In this paper, synthetic search queries inferred from existing queries are used to acquire attributes. The queries lead to ranked lists of attributes whose accuracy is higher than with equivalent methods operating over queries. Current work investigates alternative methods for combining attributes from multiple individual runs; the expansion of the target query templates used to extract attributes; and further applications of the inferred queries, in information extraction and beyond.

References

E. Alfonseca, M. Paşca, and E. Robledo-Arnuncio. 2010. Acquisition of instance attributes via labeled and related instances. In *Proceedings of the 33rd International Conference on Research and Development in Information Retrieval (SIGIR-10)*, pages 58–65, Geneva, Switzerland.

A. Bakalov, A. Fuxman, P. Talukdar, and S. Chakrabarti. 2011. Scad: collective discovery of attribute values. In *Proceedings of the 20th World Wide Web Conference (WWW-11)*, pages 447–456, Hyderabad, India.

M. Banko and O. Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 28–36, Columbus, Ohio.

K. Bellare, P. Talukdar, G. Kumaran, F. Pereira, M. Liberman, A. McCallum, and M. Dredze. 2007. Lightly-supervised attribute extraction. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS-07). Workshop on Machine Learning for Web Search*, Whistler, British Columbia.

M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. 2008. WebTables: Exploring the power of tables on the Web. In *Proceedings of the 34th Conference on Very Large Data Bases (VLDB-08)*, pages 538–549, Auckland, New Zealand.

G. Cui, Q. Lu, W. Li, and Y. Chen. 2009. Automatic acquisition of attributes for ontology construction. In *Proceedings of the 22nd International Conference on Computer Processing of Oriental Languages*, pages 248–259, Hong Kong.

D. Davidov and A. Rappoport. 2010. Extraction and approximation of numerical attributes from the Web. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, Uppsala, Sweden.

J. Dean and S. Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI-04)*, pages 137–150, San Francisco, California.

N. Guarino. 1992. Concepts, attributes and arbitrary relations. *Data and Knowledge Engineering*, 8:249–261.

M. Heilman and N. Smith. 2010. Good question! Statistical ranking for question generation. In *Proceedings of the 2010 Conference of the North American Association for Computational Linguistics (NAACL-HLT-10)*, pages 609–617, Los Angeles, California.

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic class learning from the Web with hyponym pattern linkage graphs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 1048–1056, Columbus, Ohio.

D. Lin and P. Pantel. 2002. Concept discovery from text. In *Proceedings of the 19th International Conference on Computational linguistics (COLING-02)*, pages 1–7, Taipei, Taiwan.

D. Lin and X. Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP-09)*, pages 1030–1038, Singapore.

R. Mitkov and L. Ha. 2006. A computer-aided environment for generating multiple-choice test items. *Natural Language Engineering*, 12(2):177–194.

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

V. Nastase and M. Strube. 2008. Decoding Wikipedia categories for knowledge acquisition. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI-08)*, pages 1219–1224, Chicago, Illinois.

M. Paşca and B. Van Durme. 2007. 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.

M. Paşca, E. Alfonseca, E. Robledo-Arnuncio, R. Martin-Brualla, and K. Hall. 2010. The role of query sessions in extracting instance attributes from web search queries. In *Proceedings of the 32nd European Conference on Information Retrieval (ECIR-10)*, pages 62–74, Milton Keynes, United Kingdom.

M. Paşca. 2007. 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.

M. Paşca. 2011. Asking what no one has asked before: Using phrase similarities to generate synthetic web search queries. In *Proceedings of the 20th International Conference on Information and Knowledge Management (CIKM-11)*, Glasgow, United Kingdom.

P. Pantel, E. Crestan, A. Borkovsky, A. Popescu, and V. Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 938–947, Singapore.

K. Probst, R. Ghani, M. Krema, A. Fano, and Y. Liu. 2007. 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.

S. Raju, P. Pingali, and V. Varma. 2008. An unsupervised approach to product attribute extraction. In *Proceedings of the 31st International Conference on Research and Development in Information Retrieval (SIGIR-08)*, pages 35–42, Singapore.

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

F. Suchanek, G. Kasneci, and G. Weikum. 2007. Yago: a core of semantic knowledge unifying WordNet and Wikipedia. In *Proceedings of the 16th World Wide Web Conference (WWW-07)*, pages 697–706, Banff, Canada.

K. Tokunaga, J. Kazama, and K. Torisawa. 2005. 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.

Tom White. 2010. *Hadoop: the Definitive Guide*. O'Reilly Media, 2nd edition.

T. Wong, W. Lam, and T. Wong. 2008. An unsupervised framework for extracting and normalizing product attributes from multiple Web sites. In *Proceedings of the 31st International Conference on Research and Development in Information Retrieval (SIGIR-08)*, pages 35–42, Singapore.

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

F. Wu, R. Hoffmann, and D. Weld. 2008. Information extraction from Wikipedia: Moving down the long tail. In *Proceedings of the 14th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD-08)*, pages 731–739.

N. Yoshinaga and K. Torisawa. 2007. 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.