Extraction and Approximation of Numerical Attributes from the Web

Dmitry Davidov
ICNC
The Hebrew University
Jerusalem, Israel
dmitry@alice.nc.huji.ac.il

Ari Rappoport
Institute of Computer Science
The Hebrew University
Jerusalem, Israel
arir@cs.huji.ac.il

Abstract

We present a novel framework for automated extraction and approximation of numerical object attributes such as height and weight from the Web. Given an object-attribute pair, we discover and analyze attribute information for a set of comparable objects in order to infer the desired value. This allows us to approximate the desired numerical values even when no exact values can be found in the text.

Our framework makes use of relation defining patterns and WordNet similarity information. First, we obtain from the Web and WordNet a list of terms similar to the given object. Then we retrieve attribute values for each term in this list, and information that allows us to compare different objects in the list and to infer the attribute value range. Finally, we combine the retrieved data for all terms from the list to select or approximate the requested value.

We evaluate our method using automated question answering, WordNet enrichment, and comparison with answers given in Wikipedia and by leading search engines. In all of these, our framework provides a significant improvement.

1 Introduction

Information on various numerical properties of physical objects, such as length, width and weight is fundamental in question answering frameworks and for answering search engine queries. While in some cases manual annotation of objects with numerical properties is possible, it is a hard and labor intensive task, and is impractical for dealing with the vast amount of objects of interest. Hence, there is a need for automated semantic acquisition algorithms targeting such properties.

In addition to answering direct questions, the ability to make a crude comparison or estimation of object attributes is important as well. For example, it allows to disambiguate relationships between objects such as *X part-of Y* or *X inside Y*. Thus, a coarse approximation of the height of a house and a window is sufficient to decide that in the ‘house window’ nominal compound, ‘window’ is very likely to be a part of house and not vice versa. Such relationship information can, in turn, help summarization, machine translation or textual entailment tasks.

Due to the importance of relationship and attribute acquisition in NLP, numerous methods were proposed for extraction of various lexical relationships and attributes from text. Some of these methods can be successfully used for extracting numerical attributes. However, numerical attribute extraction is substantially different in two aspects, verification and approximation.

First, unlike most general lexical attributes, numerical attribute values are comparable. It usually makes no sense to compare the names of two actors, but it is meaningful to compare their ages. The ability to compare values of different objects allows to improve attribute extraction precision by verifying consistency with attributes of other similar objects. For example, suppose that for Toyota Corolla width we found two different values, 1.695m and 27cm. The second value can be either an extraction error or a length of a toy car. Extracting and looking at width values for different car brands and for ‘cars’ in general we find:

- Boundaries: Maximal car width is 2.195m, minimal is 88cm.
- Average: Estimated avg. car width is 1.7m.
- Direct/indirect comparisons: Toyota Corolla is wider than Toyota Corona.
- Distribution: Car width is distributed normally around the average.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1308–1317,
Uppsala, Sweden, 11-16 July 2010.

Usage of all this knowledge allows us to select the correct value of 1.695m and reject other values. Thus we can increase the precision of value extraction by finding and analyzing an entire group of comparable objects.

Second, while it is usually meaningless and impossible to approximate general lexical attribute values like an actor's name, numerical attributes can be estimated *even if they are not explicitly mentioned in the text.*

In general, attribute extraction frameworks usually attempt to discover a single correct value (e.g., capital city of a country) or a set of distinct correct values (e.g., actors of a movie). So there is essentially nothing to do when there is no explicit information present in the text for a given object and an attribute. In contrast, in numerical attribute extraction it is possible to provide an approximation even when no explicit information is present in the text, by using values of comparable objects for which information is provided.

In this paper we present a pattern-based framework that takes advantage of the properties of similar objects to improve extraction precision and allow approximation of requested numerical object properties. Our framework comprises three main stages. First, given an object name we utilize WordNet and pattern-based extraction to find a list of similar objects and their category labels. Second, we utilize a predefined set of lexical patterns in order to extract attribute values of these objects and available comparison/boundary information. Finally, we analyze the obtained information and select or approximate the attribute value for the given (object, attribute) pair.

We performed a thorough evaluation using three different applications: Question Answering (QA), WordNet (WN) enrichment, and comparison with Wikipedia and answers provided by leading search engines. QA evaluation was based on a designed dataset of 1250 questions on size, height, width, weight, and depth, for which we created a gold standard and compared against it automatically[1].

For WN enrichment evaluation, our framework discovered size and weight values for 300 WN physical objects, and the quality of results was evaluated by human judges. For interactive search, we compared our results to information obtained through Wikipedia, Google and Wolfram Alpha. Utilization of information about comparable objects provided a significant boost to numerical attribute extraction quality, and allowed a meaningful approximation of missing attribute values.

Section 2 discusses related work, Section 3 details the algorithmic framework, Section 4 describes the experimental setup, and Section 5 presents our results.

2 Related work

Numerous methods have been developed for extraction of diverse semantic relationships from text. While several studies propose relationship identification methods using distributional analysis of feature vectors (Turney, 2005), the majority of the proposed open-domain relations extraction frameworks utilize lexical patterns connecting a pair of related terms. (Hearst, 1992) manually designed lexico-syntactic patterns for extracting hypernymy relations. (Berland and Charniak, 1999; Girju et al, 2006) proposed a set of patterns for meronymy relations. Davidov and Rappoport (2008a) used pattern clusters to disambiguate nominal compound relations. Extensive frameworks were proposed for iterative discovery of any pre-specified (e.g., (Riloff and Jones, 1999; Chklovski and Pantel, 2004)) and unspecified (e.g., (Banko et al., 2007; Rosenfeld and Feldman, 2007; Davidov and Rappoport, 2008b)) relation types.

The majority of the above methods utilize the following basic strategy. Given (or discovering automatically) a set of patterns or relationship-representing term pairs, these methods mine the web for these patterns and pairs, iteratively obtaining more instances. The proposed strategies generally include some weighting/frequency/context-based algorithms (e.g. (Pantel and Pennacchiotti, 2006)) to reduce noise. Some of the methods are suitable for retrieval of numerical attributes. However, most of them do not exploit the numerical nature of the attribute data.

Our research is related to a sub-domain of question answering (Prager, 2006), since one of the applications of our framework is answering questions on numerical values. The majority of the proposed QA frameworks rely on pattern-based relationship acquisition (Ravichandran and Hovy, 2009). However, most QA studies focus on different types of problems than our paper, including question classification, paraphrasing, etc.

[1] This dataset is available in the authors' websites for the research community.

Several recent studies directly target the acquisition of numerical attributes from the Web and attempt to deal with ambiguity and noise of the retrieved attribute values. (Aramaki et al., 2007) utilize a small set of patterns to extract physical object sizes and use the averages of the obtained values for a noun compound classification task. (Banerjee et al, 2009) developed a method for dealing with quantity consensus queries (QCQs) where there is uncertainty about the answer quantity (e.g. “driving time from Paris to Nice”). They utilize a textual snippet feature and snippet quantity in order to select and rank intervals of the requested values. This approach is particularly useful when it is possible to obtain a substantial amount of a desired attribute values for the requested query. (Moriceau, 2006) proposed a rule-based system which analyzes the variation of the extracted numerical attribute values using information in the textual context of these values.

A significant body of recent research deals with extraction of various data from web tables and lists (e.g., (Cafarella et al., 2008; Crestan and Pantel, 2010)). While in the current research we do not utilize this type of information, incorporation of the numerical data extracted from semi-structured web pages can be extremely beneficial for our framework.

All of the above numerical attribute extraction systems utilize only direct information available in the discovered object-attribute co-occurrences and their contexts. However, as we show, indirect information available for comparable objects can contribute significantly to the selection of the obtained values. Using such indirect information is particularly important when only a modest amount of values can be obtained for the desired object. Also, since the above studies utilize only explicitly available information they were unable to approximate object values in cases where no explicit information was found.

3 The Attribute Mining Framework

Our algorithm is given an object and an attribute. In the WN enrichment scenario, it is also given the object’s synset. The algorithm comprises three main stages: (1) mining for similar objects and determination of a class label; (2) mining for attribute values and comparison statements; (3) processing the results.

3.1 Similar objects and class label

To verify and estimate attribute values for the given object we utilize similar objects (co-hyponyms) and the object’s class label (hypernym). In the WN enrichment scenario we can easily obtain these, since we get the object’s synset as input. However, in Question Answering (QA) scenarios we do not have such information. To obtain it we employ a strategy which uses WordNet along with pattern-based web mining.

Our web mining part follows common pattern-based retrieval practice (Davidov et al., 2007). We utilize Yahoo! Boss API to perform search engine queries. For an object name Obj we query the Web using a small set of pre-defined co-hyponymy patterns like *“as * and/or [Obj]”*[2]. In the WN enrichment scenario, we can add the WN class label to each query in order to restrict results to the desired word sense. In the QA scenario, if we are given the full question and not just the (object, attribute) pair we can add terms appearing in the question and having a strong PMI with the object (this can be estimated using any fixed corpus). However, this is not essential.

We then extract new terms from the retrieved web snippets and use these terms iteratively to retrieve more terms from the Web. For example, when searching for an object ‘Toyota’, we execute a search engine query *[“as * and Toyota”]* and we might retrieve a text snippet containing *“. . . as Honda and Toyota . . . ”*. We then extract from this snippet the additional word ‘Honda’ and use it for iterative retrieval of additional similar terms. We attempt to avoid runaway feedback loop by requiring each newly detected term to co-appear with the original term in at least a single co-hyponymy pattern.

WN class labels are used later for the retrieval of boundary values, and here for expansion of the similar object set. In the WN enrichment scenario, we already have the class label of the object. In the QA scenario, we automatically find class labels as follows. We compute for each WN subtree a coverage value, the number of retrieved terms found in the subtree divided by the number of subtree terms, and select the subtree having the highest coverage. In all scenarios, we add all terms found in this subtree to the retrieved term list. If no WN subtree with significant (> 0.1) coverage is found,

[2] “*” means a search engine wildcard. Square brackets indicate filled slots and are not part of the query.

we retrieve a set of category labels from the Web using hypernymy detection patterns like *"* such as [Obj]"* (Hearst, 1992). If several label candidates were found, we select the most frequent.

Note that we perform this stage only once for each object and do not need to repeat it for different attribute types.

3.2 Querying for values, bounds and comparison data

Now we would like to extract the attribute values for the given object and its similar objects. We will also extract bounds and comparison information in order to verify the extracted values and to approximate the missing ones.

To allow us to extract attribute-specific information, we provided the system with a seed set of extraction patterns for each attribute type. There are three kinds of patterns: value extraction, bounds and comparison patterns. We used up to 10 patterns of each kind. These patterns are the only attribute-specific resource in our framework.

Value extraction. The first pattern group, P_{values}, allows extraction of the attribute values from the Web. All seed patterns of this group contain a measurement unit name, attribute name, and some additional anchoring words, e.g., *'Obj is * [height unit] tall'* or *'Obj width is * [width unit]'*. As in Section 3.1, we execute search engine queries and collect a set of numerical values for each pattern. We extend this group iteratively from the given seed as commonly done in pattern-based acquisition methods. To do this we re-query the Web with the obtained (object, attribute value, attribute name) triplets (e.g., *'[Toyota width 1.695m]'*). We then extract new patterns from the retrieved search engine snippets and re-query the Web with the new patterns to obtain more attribute values.

We provided the framework with unit names and with an appropriate conversion table which allows to convert between different measurement systems and scales. The provided names include common abbreviations like *cm/centimeter*. All value acquisition patterns include unit names, so we know the units of each extracted value. At the end of the value extraction stage, we convert all values to a single unit format for comparison.

Boundary extraction. The second group, $P_{boundary}$, consists of boundary-detection patterns like *'the widest [label] is * [width unit]'*. These patterns incorporate the class labels discovered in the previous stage. They allow us to find maximal and minimal values for the object category defined by labels. If we get several lower bounds and several upper bounds, we select the highest upper bound and the lowest lower bound.

Extraction of comparison information. The third group, $P_{compare}$, consists of comparison patterns. They allow to compare objects directly even when no attribute values are mentioned. This group includes attribute equality patterns such as *'[Object1] has the same width as [Object2]'*, and attribute inequality ones such as *'[Object1] is wider than [Object2]'*. We execute search queries for each of these patterns, and extract a set of ordered term pairs, keeping track of the relationships encoded by the pairs.

We use these pairs to build a directed graph (Widdows and Dorow, 2002; Davidov and Rappoport, 2006) in which nodes are objects (not necessarily with assigned values) and edges correspond to extracted co-appearances of objects inside the comparison patterns. The directions of edges are determined by the comparison sign. If two objects co-appear inside an equality pattern we put a bidirectional edge between them.

3.3 Processing the collected data

As a result of the information collection stage, for each object and attribute type we get:

- A set of attribute values for the requested object.
- A set of objects similar or comparable to the requested object, some of them annotated with one or many attribute values.
- Upper and lowed bounds on attribute values for the given object category.
- A comparison graph connecting some of the retrieved objects by comparison edges.

Obviously, some of these components may be missing or noisy. Now we combine these information sources to select a single attribute value for the requested object or to approximate this value. First we apply bounds, removing out-of-range values, then we use comparisons to remove inconsistent comparisons. Finally we examine the remaining values and the comparison graph.

Processing bounds. First we verify that indeed most ($\geq 50\%$) of the retrieved values fit the retrieved bounds. If the lower and/or upper bound contradicts more than half of the data, we reject the bound. Otherwise we remove all values which do not satisfy one or both of the accepted bounds. If no bounds are found or if we disable the bound retrieval (see Section 4.1), we assign the maximal and minimal observed values as bounds.

Since our goal is to obtain a value for the single requested object, if at the end of this stage we remain with a single value, no further processing is needed. However, if we obtain a set of values or no values at all, we have to utilize comparison data to select one of the retrieved values or to approximate the value in case we do not have an exact answer.

Processing comparisons. First we simplify the comparison graph. We drop all graph components that are not connected (when viewing the graph as undirected) to the desired object.

Now we refine the graph. Note that each graph node may have a single value, many assigned values, or no assigned values. We define *assigned nodes* as nodes that have at least one value. For each directed edge $E(A \rightarrow B)$, if both A and B are assigned nodes, we check if $Avg(A) \leq Avg(B)$[3]. If the average values violate the equation, we gradually remove up to half of the highest values for A and up to half of the lowest values for B till the equation is satisfied. If this cannot be done, we drop the edge. We repeat this process until every edge that connects two assigned nodes satisfies the inequality.

Selecting an exact attribute value. The goal now is to select an attribute value for the given object. During the first stage it is possible that we directly extract from the text a set of values for the requested object. The bounds processing step rejects some of these values, and the comparisons step may reject some more. If we still have several values remaining, we choose the most frequent value based on the number of web snippets retrieved during the value acquisition stage. If there are several values with the same frequency we select the median of these values.

Approximating the attribute value. In the case when we do not have any values remaining after the bounds processing step, the object node will remain unassigned after construction of the comparison graph, and we would like to estimate its value. Here we present an algorithm which allows us to set the values of all unassigned nodes, including the node of the requested object.

In the algorithm below we treat all node groups connected by bidirectional (equality) edges as a same-value group, i.e., if a value is assigned to one node in the group, the same value is immediately assigned to the rest of the nodes in the same group.

We start with some preprocessing. We create dummy lower and upper bound nodes L and U with corresponding upper/lower bound values obtained during the previous stage. These dummy nodes will be used when we encounter a graph which ends with one or more nodes with no available numerical information. We then connect them to the graph as follows: (1) if A has no incoming edges, we add an edge $L \rightarrow A$; (2) if A has no outgoing edges, we add an edge $A \rightarrow U$.

We define a *legal unassigned path* as a directed path $A_0 \rightarrow A_1 \rightarrow \ldots \rightarrow A_n \rightarrow A_{n+1}$ where A_0 and A_{n+1} are assigned satisfying $Avg(A_0) \leq Avg(A_{n+1})$ and $A_1 \ldots A_n$ are unassigned. We would like to use dummy bound nodes only in cases when no other information is available. Hence we consider paths $L \rightarrow \ldots \rightarrow U$ connecting both bounds are illegal. First we assign values for all unassigned nodes that belong to a single legal unassigned path, using a simple linear combination:
$Val(A_i)_{i \in (1 \ldots n)} =$

$$\frac{n+1-i}{n+1} Avg(A_0) + \frac{i}{n+1} Avg(A_{n+1})$$

Then, for all unassigned nodes that belong to multiple legal unassigned paths, we compute node value as above for each path separately and assign to the node the average of the computed values.

Finally we assign the average of all extracted values within bounds to all the remaining unassigned nodes. Note that if we have no comparison information and no value information for the requested object, the requested object will receive the average of the extracted values of the whole set of the retrieved comparable objects and the comparison step will be essentially empty.

4 Experimental Setup

We performed automated question answering (QA) evaluation, human-based WN enrichment evaluation, and human-based comparison of our results to data available through Wikipedia and to the top results of leading search engines.

[3] Avg. is of values of an object, without similar objects.

4.1 Experimental conditions

In order to test the main system components, we ran our framework under five different conditions:

- **FULL:** All system components were used.
- **DIRECT:** Only direct pattern-based acquisition of attribute values (Section 3.2, value extraction) for the given object was used, as done in most general-purpose attribute acquisition systems. If several values were extracted, the most common value was used as an answer.
- **NOCB:** No boundary and no comparison data were collected and processed ($P_{compare}$ and P_{bounds} were empty). We only collected and processed a set of values for the similar objects.
- **NOB:** As in **FULL** but no boundary data was collected and processed (P_{bounds} was empty).
- **NOC:** As in **FULL** but no comparison data was collected and processed ($P_{compare}$ was empty).

4.2 Automated QA Evaluation

We created two QA datasets, Web and TREC based.

Web-based QA dataset. We created QA datasets for size, height, width, weight, and depth attributes. For each attribute we extracted from the Web 250 questions in the following way. First, we collected several thousand questions, querying for the following patterns: "How long/tall/wide/heavy/deep/high is","What is the size/width/height/depth/weight of". Then we manually filtered out non-questions and heavily context-specific questions, e.g., "*what is the width of the triangle*". Next, we retained only a single question for each entity by removing duplicates.

For each of the extracted questions we manually assigned a gold standard answer using trusted resources including books and reliable Web data. For some questions, the exact answer is the only possible one (e.g., the height of a person), while for others it is only the center of a distribution (e.g., the weight of a coffee cup). Questions with no trusted and exact answers were eliminated. From the remaining questions we randomly selected 250 questions for each attribute.

TREC-based QA dataset. As a small complementary dataset we used relevant questions from the TREC Question Answering Track 1999-2007. From 4355 questions found in this set we collected 55 (17 size, 2 weight, 3 width, 3 depth and 30 height) questions.

Examples. Some example questions from our datasets are (correct answers are in parentheses): How tall is Michelle Obama? (180cm); How tall is the tallest penguin? (122cm); What is the height of a tennis net? (92cm); What is the depth of the Nile river? (1000cm = 10 meters); How heavy is a cup of coffee? (360gr); How heavy is a giraffe? (1360000gr = 1360kg); What is the width of a DNA molecule? (2e-7cm); What is the width of a cow? (65cm).

Evaluation protocol. Evaluation against the datasets was done automatically. For each question and each condition our framework returned a numerical value marked as either an exact answer or as an approximation. In cases where no data was found for an approximation (no similar objects with values were found), our framework returned no answer.

We computed precision[4], comparing results to the gold standard. Approximate answers are considered to be correct if the approximation is within 10% of the gold standard value. While a choice of 10% may be too strict for some applications and too generous for others, it still allows to estimate the quality of our framework.

4.3 WN enrichment evaluation

We manually selected 300 WN entities from about 1000 randomly selected objects below the object tree in WN, by filtering out entities that clearly do not possess any of the addressed numerical attributes.

Evaluation was done using human subjects. It is difficult to do an automated evaluation, since the nature of the data is different from that of the QA dataset. Most of the questions asked over the Web target named entities like specific car brands, places and actors. There is usually little or no variability in attribute values of such objects, and the major source of extraction errors is name ambiguity of the requested objects.

WordNet physical objects, in contrast, are much less specific and their attributes such as size and

[4] Due to the nature of the task recall/f-score measures are redundant here

weight rarely have a single correct value, but usually possess an acceptable numerical range. For example, the majority of the selected objects like ‘apple’ are too general to assign an exact size. Also, it is unclear how to define acceptable values and an approximation range. Crudeness of desired approximation depends both on potential applications and on object type. Some objects show much greater variability in size (and hence a greater range of acceptable approximations) than others. This property of the dataset makes it difficult to provide a meaningful gold standard for the evaluation. Hence in order to estimate the quality of our results we turn to an evaluation based on human judges.

In this evaluation we use only *approximate* retrieved values, keeping out the small amount of returned *exact* values[5].

We have mixed (Object, Attribute name, Attribute value) triplets obtained through each of the conditions, and asked human subjects to assign these to one of the following categories:

- The attribute value is reasonable for the given object.
- The value is a very crude approximation of the given object attribute.
- The value is incorrect or clearly misleading.
- The object is not familiar enough to me so I cannot answer the question.

Each evaluator was provided with a random sample of 40 triplets. In addition we mixed in 5 manually created clearly correct triplets and 5 clearly incorrect ones. We used five subjects, and the agreement (inter-annotator Kappa) on shared evaluated triplets was 0.72.

4.4 Comparisons to search engine output

Recently there has been a significant improvement both in the quality of search engine results and in the creation of manual well-organized and annotated databases such as Wikipedia.

Google and Yahoo! queries frequently provide attribute values in the top snippets or in search result web pages. Many Wikipedia articles include infoboxes with well-organized attribute values. Recently, the Wolfram Alpha computational knowledge engine presented the computation of attribute values from a given query text.

Hence it is important to test how well our framework can complement the manual extraction of attributes from resources such as Wikipedia and top Google snippets. In order to test this, we randomly selected 100 object-attribute pairs from our Web QA and WordNet datasets and used human subjects to test the following:

1. **Go1**: Querying Google for *[object-name attribute-name]* gives in some of the first three snippets a correct value or a good approximation value[6] for this pair.
2. **Go2**: Querying Google for *[object-name attribute-name]* and following the first three links gives a correct value or a good approximation value.
3. **Wi**: There is a Wikipedia page for the given object and it contains an appropriate attribute value or an approximation in an infobox.
4. **Wf**: A Wolfram Alpha query for *[object-name attribute-name]* retrieves a correct value or a good approximation value

5 Results

5.1 QA results

We applied our framework to the above QA datasets. Table 1 shows the precision and the percentage of approximations and exact answers.

Looking at %Exact+%Approx, we can see that for all datasets only 1-9% of the questions remain unanswered, while correct exact answers are found for 65%/87% of the questions for Web/TREC (% Exact and Prec(Exact) in the table). Thus approximation allows us to answer 13-24% of the requested values which are either simply missing from the retrieved text or cannot be detected using the current pattern-based framework. Comparing performance of FULL to DIRECT, we see that our framework not only allows an approximation when no exact answer can be found, but also significantly increases the precision of exact answers using the comparison and the boundary information. It is also apparent that both boundary and comparison features are needed to achieve good performance and that using both of them achieves substantially better results than each of them separately.

[5] So our results are in fact higher than shown.

[6] As defined in the human subject questionnaire.

	FULL	DIRECT	NOCB	NOB	NOC
Web QA					
Size					
%Exact	80	82	82	82	80
Prec(Exact)	76	40	40	54	65
%Approx	16	-	14	14	16
Prec(Appr)	64	-	34	53	46
Height					
%Exact	79	84	84	84	79
Prec(Exact)	86	56	56	69	70
%Approx	16	-	11	11	16
Prec(Appr)	72	-	25	65	53
Width					
%Exact	74	76	76	76	74
Prec(Exact)	86	45	45	60	72
%Approx	17	-	15	15	17
Prec(Appr)	75	-	26	63	55
Weight					
%Exact	71	73	73	73	71
Prec(Exact)	82	57	57	64	70
Prec(Appr)	24	-	22	22	24
%Approx	61	-	39	51	46
Depth					
%Exact	82	82	82	82	82
Prec(Exact)	89	60	60	71	78
%Approx	19	-	19	19	19
Prec(Appr)	92	-	58	76	63
Total average					
%Exact	77	79	79	79	77
Prec(Exact)	84	52	52	64	71
%Approx	18	-	16	16	19
Prec(Appr)	72	-	36	62	53
TREC QA					
%Exact	87	90	90	90	87
Prec(Exact)	**100**	62	62	84	76
%Approx	13	-	9	9	13
Prec(Appr)	**57**	-	20	40	57

Table 1: Precision and amount of exact and approximate answers for QA datasets.

Comparing results for different question types we can see substantial performance differences between the attribute types. Thus depth shows much better overall results than width. This is likely due to a lesser difficulty of depth questions or to a more exact nature of available depth information compared to width or size.

5.2 WN enrichment

As shown in Table 2, for the majority of examined WN objects, the algorithm returned an approximate value, and only for 13-15% of the objects (vs. 70-80% in QA data) the algorithm could retrieve exact answers.

Note that the common pattern-based acquisition framework, presented as the DIRECT condition, could only extract attribute values for 15% of the objects since it does not allow approximations and may only extract values from the text where they explicitly appear.

	FULL	DIRECT	NOCB	NOB	NOC
Size					
%Exact	15.3	18.0	18.0	18.0	15.3
%Approx	80.3	-	38.2	20.0	23.6
Weight					
%Exact	11.8	12.5	12.5	12.5	11.8
%Approx	71.7	-	38.2	20.0	23.6

Table 2: Percentage of exact and approximate values for the WordNet enrichment dataset.

	FULL	NOCB	NOB	NOC
Size				
%Correct	73	21	49	28
%Crude	15	54	31	49
%Incorrect	8	21	16	19
Weight				
%Correct	64	24	46	38
%Crude	24	45	30	41
%Incorrect	6	25	18	15

Table 3: Human evaluation of approximations for the WN enrichment dataset (the percentages are averaged over the human subjects).

Table 3 shows human evaluation results. We see that the majority of approximate values were clearly accepted by human subjects, and only 6-8% were found to be incorrect. We also observe that both boundary and comparison data significantly improve the approximation results. Note that DIRECT is missing from this table since no approximations are possible in this condition.

Some examples for WN objects and approximate values discovered by the algorithm are: Sandfish, 15gr; skull, 1100gr; pilot, 80.25kg. The latter value is amusing due to the high variability of the value. However, even this value is valuable, as a sanity check measure for automated inference systems and for various NLP tasks (e.g., 'pilot jacket' likely refers to a jacket used by pilots and not vice versa).

5.3 Comparison with search engines and Wikipedia

Table 4 shows results for the above datasets in comparison to the proportion of correct results and the approximations returned by our framework under the FULL condition (correct exact values and approximations are taken together).

We can see that our framework, due to its approximation capability, currently shows significantly greater coverage than manual extraction of data from Wikipedia infoboxes or from the first

	FULL	Go1	Go2	Wi	Wf
Web QA	83	32	40	15	21
WordNet	87	24	27	18	5

Table 4: Comparison of our attribute extraction framework to manual extraction using Wikipedia and search engines.

search engine results.

6 Conclusion

We presented a novel framework which allows an automated extraction and approximation of numerical attributes from the Web, even when no explicit attribute values can be found in the text for the given object. Our framework retrieves similarity, boundary and comparison information for objects similar to the desired object, and combines this information to approximate the desired attribute.

While in this study we explored only several specific numerical attributes like size and weight, our framework can be easily augmented to work with any other consistent and comparable attribute type. The only change required for incorporation of a new attribute type is the development of attribute-specific $P_{boundary}$, P_{values}, and $P_{compare}$ pattern groups; the rest of the system remains unchanged.

In our evaluation we showed that our framework achieves good results and significantly outperforms the baseline commonly used for general lexical attribute retrieval[7].

While there is a growing justification to rely on extensive manually created resources such as Wikipedia, we have shown that in our case automated numerical attribute acquisition could be a preferable option and provides excellent coverage in comparison to handcrafted resources or manual examination of the leading search engine results. Hence a promising direction would be to use our approach in combination with Wikipedia data and with additional manually created attribute rich sources such as Web tables, to achieve the best possible performance and coverage.

We would also like to explore the incorporation of approximate discovered numerical attribute data into existing NLP tasks such as noun compound classification and textual entailment.

[7] It should be noted, however, that in our DIRECT baseline we used a basic pattern-based retrieval strategy; more sophisticated strategies for value selection might bring better results.

References

Eiji Aramaki, Takeshi Imai, Kengo Miyo and Kazuhiko Ohe. 2007 UTH: SVM-based Semantic Relation Classification using Physical Sizes. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007).*

Somnath Banerjee, Soumen Chakrabarti and Ganesh Ramakrishnan. 2009. Learning to Rank for Quantity Consensus Queries. *SIGIR '09.*

Michele Banko, Michael J Cafarella , Stephen Soderland, Matt Broadhead and Oren Etzioni. 2007. Open information extraction from the Web. *IJCAI '07.*

Matthew Berland, Eugene Charniak, 1999. Finding parts in very large corpora. *ACL '99.*

Michael Cafarella, Alon Halevy, Yang Zhang, Daisy Zhe Wang and Eugene Wu. 2008. WebTables: Exploring the Power of Tables on the Web. *VLDB '08.*

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: mining the Web for fine-grained semantic verb relations. *EMNLP '04.*

Eric Crestan and Patrick Pantel. 2010. Web-Scale Knowledge Extraction from Semi-Structured Tables. *WWW '10.*

Dmitry Davidov and Ari Rappoport. 2006. Efficient Unsupervised Discovery of Word Categories Using Symmetric Patterns and High Frequency Words. *ACL-Coling '06.*

Dmitry Davidov, Ari Rappoport and Moshe Koppel. 2007. Fully unsupervised discovery of concept-specific relationships by web mining. *ACL '07.*

Dmitry Davidov and Ari Rappoport. 2008a. Classification of Semantic Relationships between Nominals Using Pattern Clusters. *ACL '08.*

Dmitry Davidov and Ari Rappoport. 2008b. Unsupervised Discovery of Generic Relationships Using Pattern Clusters and its Evaluation by Automatically Generated SAT Analogy Questions. *ACL '08.*

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2006. Automatic discovery of part-whole relations. *Computational Linguistics,* 32(1).

Marty Hearst, 1992. Automatic acquisition of hyponyms from large text corpora. *COLING '92.*

Veronique Moriceau, 2006. Numerical Data Integration for Cooperative Question-Answering. *EACL - KRAQ06 '06.*

John Prager, 2006. Open-domain question-answering. *In Foundations and Trends in Information Retrieval,vol. 1, pp 91-231.*

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: leveraging generic patterns for automatically harvesting semantic relations. *COLING-ACL '06*.

Deepak Ravichandran and Eduard Hovy. 2002 Learning Surface Text Patterns for a Question Answering System. *ACL '02*.

Ellen Riloff and Rosie Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. *AAAI '99*.

Benjamin Rosenfeld and Ronen Feldman. 2007. Clustering for unsupervised relation identification. *CIKM '07*.

Peter Turney, 2005. Measuring semantic similarity by latent relational analysis, *IJCAI '05*.

Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised Lexical acquisition. *COLING '02*.