entropy

MDPI

Article

A *PUT*-Based Approach to Automatically Extracting Quantities and Generating Final Answers for Numerical Attributes

Yaqing Liu [1,*], Lidong Wang [2], Rong Chen [1], Yingjie Song [3] and Yalin Cai [1]

1 School of Information Science & Technology, Dalian Maritime University, Dalian 116026, China; rchen@dlmu.edu.cn (R.C.); cyllc2015@163.com (Y.C.)
2 Department of Mathematics, Dalian Maritime University, Dalian 116026, China; ldwang@hotmail.com
3 School of Computer Science and Technology, Shandong Institute of Business and Technology, Yantai 264005, China; tiantianyingjie@gmail.com
* Correspondence: liuyaqing@dlmu.edu.cn; Tel.: +86-411-8472-3669

Academic Editor: Andreas Holzinger
Received: 14 January 2016; Accepted: 12 June 2016; Published: 22 June 2016

Abstract: Automatically extracting quantities and generating final answers for numerical attributes is very useful in many occasions, including question answering, image processing, human-computer interaction, *etc.* A common approach is to learn linguistics templates or wrappers and employ some algorithm or model to generate a final answer. However, building linguistics templates or wrappers is a tough task for builders. In addition, linguistics templates or wrappers are domain-dependent. To make the builder escape from building linguistics templates or wrappers, we propose a new approach to final answer generation based on Predicates-Units Table (*PUT*), a mini domain-independent knowledge base. It is deserved to point out that, in the following cases, quantities are not represented well. Quantities are absent of units. Quantities are perhaps wrong for a given question. Even if all of them are represented well, their units are perhaps inconsistent. These cases have a strong impact on final answer solving. One thousand nine hundred twenty-six real queries are employed to test the proposed method, and the experimental results show that the average correctness ratio of our approach is 87.1%.

Keywords: numerical attribute; information extraction; quantity; entity extraction

1. Introduction

Quantity extraction for numerical attribute is very useful in many occasions including question answering [1], image processing [2], human-computer interaction [3], *etc.* For example, quantity extraction is necessary to final answer solving for a numerical question in question answering. Extracted quantity (e.g., size of a physical object) is helpful to distinguish physical objects in image processing. In addition, extracted quantity is helpful to find incorrect input in human-computer interaction. However, it is a hard and intensive work to extract a quantity manually. Hence, automatically extracting quantities and generating final answers for numerical attributes is emphasized.

For general entity extraction, a common approach to final answer generation is to extract the candidate entities from information sources. Consequently, candidate entities are classified by semantic similarity. At last, the most frequent candidate entity is selected as the final answer. For numerical attributes, the process of final answer generation is a little different. If there is not the most frequent quantity, an alternative approach is to calculate the average value of all quantities as the final answer. However, they are not sometimes represented well. Some are absent of units. Some are perhaps wrong for a given question. Even if all of them are represented well, their units are perhaps inconsistent.

These exceptions have a strong impact on final answer generation. An example is "*What is the length of an automobile?*" The quantity set is {4420, 166.25, 114, 106, 114, 5.0165, 4.19608, 3.9116, 4.93014}. Each quantity is absent of units. One hundred fourteen is selected as the final answer because it is the most frequent quantity. Obviously, the final answer is unmeaning because the unit is absent. Another example is "*What is the weight of a dog?*" The quantity set is {(2.0, kg), (2.5, kg), (10.0, kg), (50.0, cm), . . . }. Obviously, (50, cm) is a wrong quantity and should be dropped. Since the quality of the final answer depends on quantities, the keynote of our strategy is to drop all wrong quantities, keep all correct quantities, and infer units of the quantities which are absent of units. In addition, compared to a general entity, the quantity of numerical attributes is likely used as an operand for a complex numerical question. For instance, the answer (105.4, km^2) is used as an operand to solve the question "*How many times the area of Beijing is larger than Paris?*" Obviously, accuracy of the answer of the question depends on accuracies of the operands. The more accurate are the operands, the more accurate is the answer of question. This shows that answer solving for numerical questions is worth exploring.

This paper is organized as follows: Section 2 focuses on some preliminaries; Section 3 focuses on the proposed strategy; Section 4 focuses on the experiments and evaluation; and Section 5 focuses on related work. The paper finishes in Section 6, in which conclusions are drawn.

2. Preliminaries

Definition 1. *A* ***query*** *q is 2-tuple q = (subj, pred), where subj is the subject of q and pred is the predicate of q.*

For example, q = (*earth*, *radius*) is a query that means "*What is the radius of the Earth?*", and *earth* is the subject of q. *radius* is the predicate of q.

Definition 2. *For a query q,* ***quantity set*** *CAS_q is defined as $CAS_q = \{ca_i = (value_i, unit_i) \mid 1 <= I <= \mid CAS_q \mid \}$. CAS_q is returned when q is submitted to Sindice [4], a semantic searing engine. For any $ca_i \in CAS_q$, ca_i is said to be a* ***quantity****, $value_i$ is said to be a* ***value*** *of ca_i, and $unit_i$ is said to be a* ***unit*** *of ca_i.*

For example, one of the quantities is (6371 km) when query q = (*earth*, *radius*) is submitted to Sindice. This means that the radius of the earth is 6371 kilometres. Note that quantities are sometimes absent of units for some queries. For example, one of the quantities is 1,321,851,888 when a query q = (*China*, *population*) is submitted to Sindice. To apply the same format to represent these quantities which are absent of units, we use "*count*" as the unit of the quantity which is absent of unit. For example, the quantity 1,321,851,888 is represented as (1,321,851,888, *count*).

Definition 3. ***Predicates-Units Table*** *PUT is defined as $\{PU_i = (t_i, P_i, U_i, s_i, UR_i, Eq_i) \mid 1 < = I <= \mid PUT \mid \}$. For any $PU_i = (t_i, P_i, U_i, s_i, UR_i, Eq_i) \in PUT$, PU_i is called a* ***predicates-units****. t_i is a* ***semantic type*** *of PU_i and is also the identification of PU_i. The first letter of t_i is capitalised. P_i is a set of predicates that have the common semantic type t_i. For any $p \in P_i$, t_i is a semantic type of p. U_i is a set of units that have common semantic type t_i. For any $u \in U_i$, t_i is a semantic type of u. $s_i \in U_i$ is an SI unit of PU_i. UR_i is defined as $\{(unit_j, ratio_j) \mid unit_j \in U_i, 1 <= j <= \mid U_i \mid$ and $ratio_j$ is a digital$\}$. Mathematics equations between units are given in tuple Eq. Let T be $\{t_1, t_2, \ldots, t_{\mid PUT \mid}\}$.*

A segment of *PUT* is shown in Table 1. For example, (*Distance*, {*length*, *height*, *width*, *radius*, *diameter*}, {*metre*, *millimetre*, *inch*, *foot*, *yard*, *centimetre*, *kilometre*}, *metre*, {(*metre*, 1), (*millimetre*, 1000), (*inch*, 39.37), (*foot*, 3.28), (*yard*, 1.09), (*centimetre*, 100), (*kilometre*, 0.001)}) is a predicates-units.

Definition 4. *For a quantity ca = (value, unit), t is a* ***semantic type*** *of ca if (t, P, U, s, UR, Eq) $\in$ PUT $\wedge$ unit $\in$ U holds. Otherwise, count is said to be a* ***semantic type*** *of ca.*

Definition 5. *For a set of quantities $CA = \{ca_1, ca_2, \ldots, ca_n\}$, t is* ***semantic type*** *of CA if for any two $a_i, a_j \in$ CA, the semantic types of both a_i and a_j are t.*

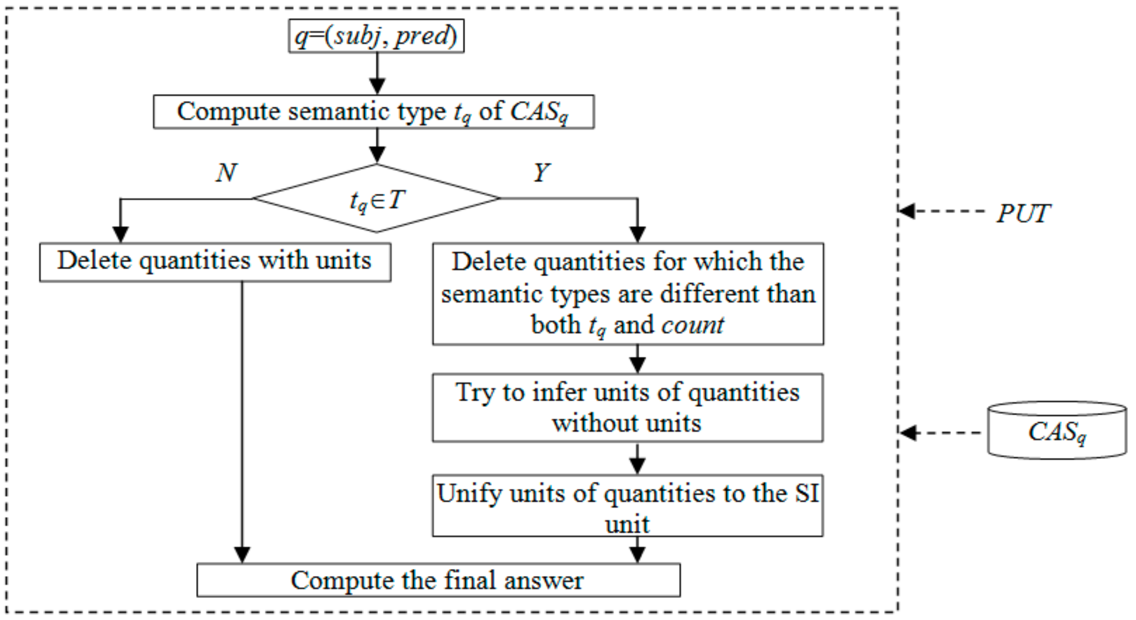
Table 1. A segment of the Predicates-Units Table.

No.	***t***	***P***	***U***	***s***	***UR***		***Eq***
1	Distance	length	metre	metre	metre	1	(1 m) = (1000 mm) = (39.37 inch) = (3.28 foot) = (1.09 yard) = (100 cm) = (0.001 km)
		height	millimetre		millimetre	1000	
		width	inch		inch	39.37	
		radius	foot		foot	3.28	
		diameter	yard		yard	1.09	
			centimetre		centimetre	100	
			kilometre		kilometre	0.001	
2	Mass	mass	kilogram	kilogram	kilogram	1	-
		weight	gram		gram	1000	(1 kg) = (1000 g) = (2.2 pound)
			pound		pound	2.2	-

3. Our Approach

The framework for final answer solving is displayed in Figure 1.

- Step 1: According to the predicate *pred* of a query *q*(*subj*, *pred*) and *PUT*, compute the semantic type t_q of *CASq*. Proceed to step 2.
- Step 2: If t_q is included in *T*, proceed to step 3. Otherwise, delete quantities with units from CAS_q and proceed to step 4.
- Step 3: Delete quantities for which the semantic types are different than both t_q and *count*. Try to infer the units of quantities which are absent of units. Unify units of quantities to the SI unit.
- Step 4: Solve the final answer. If the most common quantities occurs at least twice as often as the second most common quantity, the most common quantity is chosen as the final answer. Otherwise, if there is no obvious single quantity for a correct answer, the average value is chosen.



q=(subj, pred)
Compute semantic type t_q of CAS_q
N
Y
$t_q \in T$
PUT
Delete quantities with units
Delete quantities for which the semantic types are different than both t_q and count
Try to infer units of quantities without units
CAS_q
Unify units of quantities to the SI unit
Compute the final answer

Figure 1. The framework for final answer solving.

To make the proposed approach more comprehensible, three typical queries are introduced. The set of quantities is stated as follow.

- query 1: *q1* = (*boeing747*, *capacity*).
- query 2: *q2* = (*automobile*, *length*).
- query 3: *q3* = (*wind turbine*, *capacity*).

(1) Algorithm 1 is employed to obtain the semantic type of a quantity set. If predicate *pred* of a query *q* is found in tuple *P* of some predicates-units, the corresponding tuple *t* is returned. If predicate *pred* of *q* is not found in tuple *P* of any predicates-units, we aggregate answers in the quantity set according to their semantic type. Then, we count the quantities for every division. If the semantic type of these quantities that are in the division with the most quantities is *count*, *count* is returned. If the semantic type of these quantities that are in the division with the most quantities is not *count*, the semantic type of these quantities that are in the division with the most quantities is returned.

Semantic type *Distance* is returned because the predicate *length* of *q2* is found in *PUT*. For queries *q1* and *q3*, we divide the quantity set and count the quantities for every division because the predicate *capacity* for queries *q1* and *q3* is not found in *PUT*. For query *q1*, the number of quantities which are absent of units can be taken as 7 7. For query *q3*, the number of quantities with semantic type *Force* is 6. The number of quantities without units is 1. Thus, for query *q1*, *count* is returned. For query *q3*, semantic type *Force* is returned.

Algorithm 1. *computeSemanticTypeOfCAS*(*q*, *PUT*)

```
Input: A query $q$ = ($subj$, $pred$) and $PUT$.
Output: $t_q$, semantic type of $q$.
1.  search semantic type $p$ which matches $pred$ by $PUT$.
2.  $TS \leftarrow \Phi$
3.  $TV \leftarrow \Phi$
4.  for each ($value$, $unit$) $\in CAS_q$ do
5.  $t_u \leftarrow getT((value, unit), PUT)$ //$getT$ returns semantic type $t_u$ of ($value$, $unit$).
6.  if $t_u \notin TS$ then
7.  $TV \leftarrow TV \cup \{(t_u,\{value\})\}$
8.  $TS \leftarrow TS \cup \{t_u\}$
9.  else
10. $V \leftarrow getV(TV, t_u)$ //$getV$ returns the set $V$ which meets $\{t_u,V\} \in TV$.
11. $V \leftarrow V \cup \{v\}$
12. end if
13. end for
14. return $maxT(TV)$ //$maxT$ returns $t_m$ which meets $(t_m,V) \in TV$ and $\forall(t',V') \in TV$, $|V'| <= |V|$.
```

(2) Algorithm 2 is employed to refine the quantity set. Noisy answers are deleted from the quantity set. If the semantic type returned by Algorithm 1 is *count*, delete the quantities for which the semantic types are not *count*. Otherwise, delete the quantities for which the semantic type is not *count* or the semantic type returned by Algorithm 1. For queries *q1*, *q2*, and *q3*, the refined quantity set is displayed in Table 2.

Algorithm 2. o*ptimizeQuantities*(CAS_q, *PUT*)

```
Input: $CAS_q$ and $PUT$
Output: $CAS_q$, refined quantity set.
1.  if $computeSemanticTypeOfQuery(q, PUT)$==$count$ then
2.  $CAS_q \leftarrow optimize1(CAS_q)$ // $optimize1$ is employed to delete quantities whose semantic
                                   //types are not $count$ from $CAS_q$.
3.  else
4.  $CAS_q \leftarrow optimize2(CAS_q)$ // $optimize2$ is employed to delete answers quantities whose
        //semantic type is not count or the semantic type returned by Algorithm 1 from $CAS_q$.
5.  end if
6.  return $CAS_q$
```

Although all of CAS_{q1}, CAS_{q2}, CAS_{q3} remain unchanged, the reasons for this are different. For query *q1*, the returned semantic type is *count*. Since there are no quantities for which the semantic types are not *count*, CAS_{q1} remains unchanged. For query *q2*, the returned semantic type is *Distance*. Since the semantic types of all quantities are *count*, CAS_{q2} remains unchanged. For query *q3*, the returned semantic type is *Force*. Since the semantic types of all quantities are *count* or *Force*, CAS_{q3} remains unchanged.

Table 2. The optimized quantities CAS_{q1}, CAS_{q2}, CAS_{q3}.

CAS_{q1}		CAS_{q2}		CAS_{q3}	
Value	**Unit**	**Value**	**Unit**	**Value**	**Unit**
416	count	4420	count	3.6	megawatt
388	count	166.25	count	3.6	megawatt
388	count	114	count	3.6	megawatt
430	count	106	count	3.6	megawatt
416	count	114	count	5.5	count
416	count	5.0165	count	3,060,000	count
416	count	4.19608	count	2.3	megawatt
-	-	4.93014	count	-	-
-	-	3.9116	count	-	-

(3) Algorithms 3 and 4 are employed to infer the units of quantities which are absent of units. If the semantic types of all quantities are *count*, Algorithm 3 is employed to infer the units of quantities which are absent of units. First, we obtain tuple *UR*, which meets (q_t, *P*, *U*, *s*, *UR*, *Eq*) ∈ *PUT*. Second, we use the k-means algorithm to obtain cluster set *Cs*. Initially, the parameter "k" is set between 2 to sqrt(n), where n is the count of quantities. Consequently, we chose the optimal "k" according to DB (Davies–Bouldin) index which was proposed by Davies *et al.* [5]. Euclidean distance is exploited to calculate similarity. Third, we obtain all subsets of *UR*, each of which has ∣*Cs*∣ elements. According to Equation (1), compute the fit degree between every subset of *UR* and *Cs*. The subset of *UR* that has a maximum fit degree is used to assign the units of quantities.

Algorithm 3. *inferUnits1*(CAS_q, *PUT*)

```
Input: $CAS_q$ and $PUT$
Output: $CAS_q$, optimized quantity set.
1.  $UR \leftarrow getUR(t_q, PUT)$ // getUR returns tuple UR which meets $(t_q, P, U, s, UR) \in PUT$
2.  $Cs \leftarrow kMeans(CAS_q)$ // kMeans is employed to cluster $CAS_q$.
3.  $n \leftarrow |Cs|$
4.  Assign UR to Ms.
5.  for each $M \in Ms$ do
6.  $(score_{Cs}, M) \leftarrow getScore(Cs, M)$ // According to formula (1), getScore(Cs, M) returns M and fit
                                           // degree between Cs and M.
7.  end for
8.  $MM \leftarrow getMaxModel(Ms)$ //getMaxModel returns MM which has the maximum $score_{Cs}$.
9.  $1 \leftarrow i$
10. while $i{+}{+} <= n$ do
11. for each $(value, unit) \in cs_i$ do // $cs_i \in Cs$
12. $unit \leftarrow u_i$ // $(u_i, r_i) \in MM$
13. end for
14. end while
15. return $CAS_q$
```

$$Score = \frac{C}{V \times M} \sum_{i=1}^{V} \left(\log \frac{v_i}{C_{w_i}} \right)^2 \times \sum_{j=1}^{M} \left(\log \frac{m_j}{c_j} \right)^2 \quad (1)$$

- V is the number of values in the selected clusters. For instance of CAS_{q2}, V can be taken as 9.
- C is the total number of clusters. For instance of CAS_{q2}, C can be taken as 3.
- M is the number of values in the selected model. For instance of CAS_{q2}, M can be taken as 8 if the selected model is *Distance*.
- v_i is the value of the ith element.
- w_i is the identifier of the cluster to which the ith element belongs.
- c_j is the value of jth cluster.
- m_j is the value of the jth model's variable.

If there exists some quantities whose semantic types are *count*, Algorithm 4 is employed to infer the units of quantities which are absent of units. First, we obtain tuples UR, s and rs, which meet $(q_t, P, U, s, UR) \in PUT$ and $(s, rs) \in UR$. Second, we unify the unit of every quantity with a unit to s. Calculate the average value of the quantities with the unit s. Third, according to Equation (2), we can compute the fit degree between every quantity which is absent of a unit and the average value of quantities with unit s and then decide the unit of the quantities without units. Finally, unify the unit of every quantity with a unit to s again.

$$fd = \begin{cases} \frac{1}{a}, a > 1 \\ a, a <= 1 \end{cases}, \; a = (\frac{v}{m})/(\frac{rs}{r}) \quad (2)$$

- $(v, count) \in CAS_q$
- m is the average value of quantities with unit s_t
- $(u, r), (s, rs) \in UR_t$

Algorithm 4. *inferUnits2*(CAS_q, PUT)

Input: CAS_q and PUT
Output: CAS_q, optimized quantity set.

```
1.  $UR_t \leftarrow getUR(t_q, PUT)$ // getUR returns tuple UR which meets $(t_q, P, U, s, UR) \in PUT$
2.  $s_t \leftarrow getS(t_q, PUT)$ // getS returns tuple s which meets $(t_q, P, U, s, UR) \in PUT$
3.  $rs \leftarrow getRatioOfs(UR_t)$ // getRatioOfs return tuple rs which $(s, rs) \in UR$.
4.  $unifyUnits(CAS_q, UR_t)$ //According to $UR_t$, unify unit of every quantity with unit to $s_t$
5.  $m \leftarrow getAverageValue(CAS_q, s)$ // getAverageValue is employed to compute the average value of
                                         // quantities with unit $s_t$.
6.  for each $(value, unit) \in CAS_q$ do
7.  if $unit==count$ then
8.  for each $(u, r) \in UR$ do
9.  $UI \leftarrow (v/ave)/(r/rs)>1?\ (u,1/((v/ave)/(r/rs))):(u,(v/ave)/(r/rs))$
10. $unit \leftarrow maxU(UI)$ // maxU returns $u^*$ which meets $(u^*,i^*) \in UI$ and $\forall(u,i) \in UI, i<i^*$.
11. $UI \leftarrow \Phi$
12. end do
13. $unifyUnits(CAS_q, UR_t)$
14. return $CAS_q$
```

For quantity set CAS_{q2}, the clusters are *cluster1* = {4420}, *cluster2* = {166.25, 114, 106, 114}, and *cluster3* = {5.0165, 4.19608, 3.9116, 4.93014}. The subset of UR, which has a maximum fit degree with the cluster set, is {(millimetre, 1000), (inch, 39.37), (metre, 1)}. Hence, the units of the quantities of every cluster are "millimetre", "inch" and "metre", respectively. Unify the units of quantities to "metre". For details, please refer to Table 3.

For quantity set CAS_{q3}, the average value of the quantities with units is (3,600,000 $\times$ 4 + 2,300,000)/5 = 3,340,000 watt. The fit degree between the quantities without units and the average value is displayed in Table 5. The *UR* with a maximum degree with the quantity (306,000, *count*) is (watt, 1). Hence, the unit of quantity 3,060,000 is inferred as "watt". Similarly, the unit of quantity 5.5 is inferred as "megawatt". The units of the quantities of CAS_{q3} are unified to "watt". For details, one can refer to Tables 4 and 5.

Table 3. Quantities of CAS_{q2} after unit inference and standardization.

Old Value	Inferred Units	Revised Value	Unified Units
4420	mm	4.42	m
166.25	inch	4.22	m
114	inch	2.9	m
106	inch	2.69	m
114	inch	4.22	m
5.0165	m	5.0165	m
4.19608	m	4.19608	m
4.93014	m	4.93014	m
3.9116	m	3.9116	m

Table 4. Fit degrees of quantities of CAS_{q3} without units.

Quantities	(Megawatt, 0.000001)	(Kilowatt, 0.001)	(Horsepower, 0.00136)	(Watt, 1)
3,060,000	1.09×10^{-6}	0.001092	0.001484	0.916167665
5.5	0.607272727	0.001646707	0.001210814	1.64671×10^{-6}

Table 5. Quantities of CAS_{q3} after unit inference and standardization.

Old Value	Inferred Units	Revised Value	Unified Units
3.6	megwatt	3,600,000	watt
3.6	megwatt	3,600,000	watt
3.6	megwatt	3,600,000	watt
3.6	megwatt	3,600,000	watt
5.5	megwatt	5,500,000	watt
3060000	watt	3,600,000	watt
2.3	megwatt	2,300,000	watt

(4) Algorithm 5 illustrates how to solve the final answer. For quantities, if the most common value of quantities occurs at least twice as often as the second most common value of quantities, the most common value is selected as the final answer. Otherwise, if there is no obvious single quantity for a correct answer, the average value is chosen.

Algorithm 5. *getFinalAnswer* (CAS_q)

```
Input: $CAS_q$
Output: $answer_q$, final answer of $q$
1. $Vs \leftarrow divideCandidteAnswersByValue(CAS_q)$ // partCandidteAnswersByValue is employed to divide
                                                      //quantities by their values
2. $unit \leftarrow getS(t_q, PUT)$ // getS returns tuple $s$ which meets $(t_q, P, U, s, UR) \in PUT$
3. if $\exists V_1 \in Vs, \forall V_2 \in Vs, |V_1| >= 2|V_2|$ then
4.     $value \leftarrow getElement(V_1)$ // getElement is employed to get an element from $V_1$
5.     return $(value, unit)$
6. else
7.     $values \leftarrow getValues(CAS_q)$ // getValues returns set $values = \{value \mid (value, unit) \in CAS_q\}$
8.     return $(averageValue(values), unit)$ //averageValue returns the average value of values
9. end if
```

4. Experiments and Evaluation

4.1. Dataset Collection

We employ the Sindice search engine to collect quantities. Sindice is a lookup index over resources crawled on the Semantic Web. It allows applications to automatically locate documents containing information about a given resource. In addition, it allows resource retrieval through uniquely identifying inverse-functional properties, and offers a full-text search and index SPARQL end-points. The resources that support this particular semantic search engine include DBLP, Wikipedia article links, infoboxes, UniProt, and Geonames, *etc.* Around 26.6 million RDF documents have been indexed.

4.2. Dataset Statistics

We report on experiments with eight semantic types, which are "*Length*", "*Weight*", "*Speed*", "*Time*", "*Volume*", "*Area*", "*Power*", "*Count*", respectively. Based on the eight semantic types, we exploit Sindice, a semantic searching engine, to get quantities. For each query, the top 20 records returned by Sindice are retained. We extract quantities from each record to build a quantity set.

Finally, we collect 1926 real queries. The distribution is shown in Table 6. *UinP* is the set of queries whose predicates are found in the *unit* tuple of *PUT*. *!UinP* is the set of queries whose predicates are not found in the *unit* tuple of *PUT*. *NU* is the set of queries each of whose quantities is absent of units. *!NU* is the set of queries whose quantities are not absent of units.

Table 6. Distribution of queries.

	Distance	Weight	Speed	Time	Volume	Area	Power	Count
$UinP \cap NU$	12	12	6	0	18	30	0	0
$!UinP \cap NU$	0	0	0	0	0	0	0	218
$UinP \cap !NU$	252	256	184	212	234	288	162	0
$!UinP \cap !NU$	6	4	0	4	4	12	8	4
In total	270	272	190	216	256	330	170	222

4.3. Dataset Validation

For any of semantic types, the number in set $UinP \cap !NU$ is far more than $UinP \cap NU$, $!UinP \cap NU$, or $!UinP \cap !NU$. Hence, the whole effect of experiments heavily depends on the effect of experiments on $UinP \cap !NU$. According to Table 6, the proportion of $UinP \cap !NU$ is far higher than $UinP \cap NU$, $!UinP \cap NU$ and $!UinP \cap !NU$ for any of *Distance*, *Weight*, *Speed*, *Time*, *Volume*, *Area*, and *Power*. The proportion of $!UinP \cap NU$ is far higher than $UinP \cap NU$, $UinP \cap !NU$ and $!UinP \cap !NU$ for *Count*.

The correctness ratio of queries is used to evaluate our approach. The correctness ratio of queries is calculated according to Equation (3). Correct answers are defined according to our knowledge. A correct answer is a 3-tuples (*lb*, *ub*, *SU*), where, *lb* and *ub* are values, *SU* is a SI unit. A final answer (*v*, *u*) is said to be correct if *v* is between *lb* and *ub* and *u* is same to *SU*. Otherwise, it is said to be incorrect. For instance, the correct answer of query (*dog*, *weight*) is (1.36, 81.81, kg). If a quantity is (30.0, kg), the quantity is said to be correct. The correctness ratio of queries is shown in Table 7.

$$cr = N_c/N \tag{3}$$

- N is the number of queries.
- N_c is the number of the queries for which correct final answers are returned.

Table 7. Correctness ratio of queries.

	Distance	Weight	Speed	Time	Volume	Area	Power	Count	Average
UinP∩NU	66.7%	50.0%	33.3%	-	77.8%	73.3%	-	-	60.22%
!UinP∩NU	-	-	-	-	-	-	-	95.47%	95.47%
UinP∩!NU	95.3%	92.1%	96.2%	92.2%	97.4%	92.6%	98.1%	-	94.84%
!UinP∩!NU	100%	100%	100%	100%	100%	83.3%	100%	100%	98%
Average	87.33%	80.7%	76.5%	96.1%	91.73%	83.07%	99.05%	100%	87.1%

Figure 2 illustrates the direct-viewing chart of correctness ratio of queries. For *Count*, the correctness ratios of *!UinP∩NU* and *!UinP∩!NU* are high. Some queries fail because incorrect quantities are not dropped.

For any of *Distance*, *Weight*, *Speed*, *Time*, *Volume*, *Area*, and *Power*, the correctness ratio of *UinP∩!NU* and *!UinP∩!NU* are far higher than *UinP∩NU*. It shows that it gets better results than all of the candidate answers that are not absent of units. The correctness ratio of *UinP∩!NU* is higher than the correctness ratio of *!UinP∩!NU* because the number of *UinP∩!NU* is far more than *!UinP∩!NU*. For *UinP∩!NU* and *!UinP∩!NU*, some queries fail because incorrect quantities are not dropped. For *UinP∩NU*, there are two exceptions that do harm to the correctness ratio. The first exception is that the ratio records with a maximum fit degree are possibly more than one.

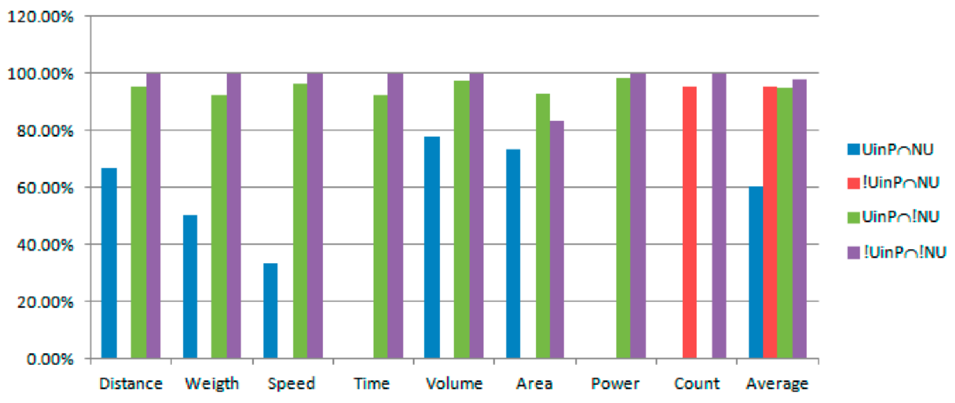


Figure 2. The direct-viewing chart of correctness ratio of queries.

For example, quantity is {521,000, 521}. By calculating the fit degree, two ratio records with a maximum fit degree are obtained from the ratio model. The first ratio record is "millimetre:metre = 1000:1". The second ratio record is "litre:cubic metre = 1000:1". Obviously, only one of the two ratio records is correct. However, the proposed approach could not distinguish between them. Another exception is that the relationship between quantities is not discerned. For query (*Russia*, *Area*), seven quantities 79,400, 16,995,800, 17,075,200, 79400, 16,995,800, 17,075,200, 560 are returned. According to our approach, the final answer is (12.049779, m^2). The correct answer is (17.035500×10^{12}, m^2). After looking up the sources of the quantities, we know that the quantity 17,075,200 is the total area of Russia. Quantity 16,995,800 is the land area of Russia. Quantity 79,400 is the water area of Russia. These data are clustered to three clusters, *Cluster1* = {16,995,800, 17,075,200, 16,995,800, 17,075,200}, *Cluster2* = {79,400, 79,400}, and *Cluster3* = {506}. According to our approach, the units of the quantities of *Cluster1* are inferred as "square millimetre". The units of the quantities of *Cluster2* are inferred as "square centimetre". The units of the quantities of *Cluster3* are inferred as "square inch". The final answer based on the proposed approach is incorrect because the semantic relationship of quantities such as "total area", "land area", and "water area" is not considered in our approach. The equation model "landarea + waterarea = totalarea" should be introduced. Quantities 1,699,580, 79,400, and 17,075,200 are fitting for the equation model "landarea + waterarea = totalarea" rather than the ratio model. It is shown that besides the ratio model, more models should be considered in our approach.

5. Related Work

There are many previous studies concerned with automatically extracting values for numerical attributes. Davidov and Rappoport presented a strategy to extract and approximate numerical attributes from the web [6]. Attribute values (range) of the given object are inferred based on attribute values of similar objects. Likewise, our approach is based on a set of quantities. However, the set of quantities in our approach has less noise because quantities are obtained only based on the given object. For similar objects, the attribute values probably have a great difference. For example, the area of Russia is far larger than that of the Netherlands. Russia is not appropriate as a similar object of Netherlands. Moriceau presented an approach to numerical answers generation which serves a Q and A system [7]. The results are that candidate answers are only displayed to users. Some comments, which are generated by a set of logical rules, are attached to candidate answers. Compared to Moriceau's work, we devote ourselves to processing candidate answers and generating a final answer. Maiya *et al.* employed a rule-based approach to extract measured information from a text document, e.g., scientific and technical documents [8]. The converted error, e.g., from PDF format to Word format, is also distinguished by their approach. Chakrabarti *et al.* aggregated snippet quantity and snippet text information from multiple information sources and proposed a statistical approach to learn to score and rank quantity intervals [9]. In addition, Chakrabarti *et al.* applied the approach to web tables [10]. Some extraction templates based on linguistics or wrappers need to be learned. Compared to extraction templates and wrappers, it takes minimal effort to build a *PUT*. Additionally, extraction templates and wrappers are domain-dependent and *PUT* is domain-independent. It is a critical task to recognize various formats of quantities in Chakrabarti' approach. Accordingly, it is the key to standardize quantities, e.g., inferring the units of quantities in our approach. Takamura and Tsujii employed a combined regression and ranking model with two types of fragmentary clues, including absolute clues and relative clues, to extract numerical attributes of physical objects [3]. The numerical attributes are extracted from absolute clues directly. Relative clues are used to infer and verify numerical attributes of physical objects. Absolute clues and relative clues are obtained from thesaurus WordNet based on linguistic patterns. Compared to Takamura and Tsujii's approach, quantities are obtained from a semantic web resource and a final answer is generated by analysing the returned quantity set. Numerical attributes are usually richer than absolute clues. Hence, our approach is suitable for a rich dataset.

A number of efforts have been made to solve the final answer in question-answering [11]. To the best of our knowledge, the process of answer solving is often divided into three phases [12]. The first phase is to retrieve quantities from information sources, such as databases and the web. Quantity retrieval is supported by some general tools such as ODBC and search engines. The second phase is to recognize the relations between two quantities. These relations are equivalence, inclusion, aggregation, and alternative [13,14]. The third phase is to decide the final answer. The common approach is to select the most frequent answers as the final answer. The result of the second phase has a significant impact on the final answer. Hence, we compare our strategy with the previous work on relation recognition.

- **Equivalence.** Generally, quantities are divided into a LOCATION category, DATE category, NUMBERIC category and text category. For the LOCATION category, DATE category, and NUMBER category, a normal format is defined [13,15]. For example, the format of DATE is defined as mm/dd/yyyy. The format of LOCATION is defined as the short form specified in the CIA World Factbook. The format of NUMBERIC is defined as a value-unit pair. Two quantities are equivalent if their values are the same after they are normalized. For the text category, the techniques for equivalence recognition include measuring the string similarity [16–18] or semantic similarity [19] between quantities. In our work, the categories of answers are limited to NUMBERIC. The format of answers is the value-unit pair. Two quantities are also the same after their values are the same.
- **Inclusion.** Quantities in the text category are perhaps connected through a hypernym or hyponymy relation in WordNet [20,21]. For example, "western Pacific" is included in "Pacific". Dalmas and Webber recognized the inclusion relation between quantities [22]. Quantities with an inclusion relation are viewed as the same answer. In our work, the inclusion relation is not recognized because the quantities are limited to the NUMBERIC category.
- **Contradiction.** For quantities in the text category, contradictory quantities are antonymous, negative or in contrast. Harabagiu *et al.* applied a maximum entropy model to detect contradictory quantities [23]. Based on linguistic features, such as factual or modal words, structural and lexical contrasts, and world knowledge, De Marneffe *et al.* used logistic regression to detect contradictory quantities [24]. In our work, contradictory quantities are not detected. However, we introduce the definition of semantic type. If the semantic type of a quantity is different from the semantic type of a query, the quantity is dropped from the quantity set.

Old approaches to questions classification are based on linguistics. Some patterns are built in advance according to a semantic dictionary, such as Wordnet, Hownet, *etc.* Similarity is computed between object questions and patterns. The most similar pattern is selected as the class of the object question. After all, natural language is extremely flexible so that patterns are difficult to meet all kinds of object questions. Recent approaches are based on machine learning. Classes, such as UIUC (University of Illinois at Urbana-Champaign) data set are usually provided in advance. Consequently, a classifier is built according to feature words which are extracted from questions. Feature words of object questions are inputted into a classifier and the classifier outputs the classes of object questions. Approaches of questions classification are displayed in Table 8. The features are divided into lexical feature [25–27] (word bags, word format, sentence length), syntax feature [25–29] (the part of speech, the question word, head word, dependency structure), and semantic features [27] (hypernym, synonyms). Syntax feature is the primary feature. Bayesian classifer [25,29], SVM (Support Vector Machine) [25–27,29], KNN (k-NearestNeighbor) [25], and neural net [27–29] are employed to class questions. SVM is more popular than other classifers. The classification effect based on multiple features is better than that based on a single feature. The classification effect of integrated classifiers is superior to a single classifier.

Table 8. Approaches of Questions Classification.

	Lexical Feature	Syntax Feature	Semantic Feature
Bayesian Classifier	[25]	[25,29]	-
SVM	[25–27]	[25–27,29]	[27]
KNN	[25]	[25]	-
Neural Net	[27]	[27–29]	[27]

6. Conclusions and Future Work

In this paper we built the Predicates-Units Table as a prior knowledge base. Based on the Predicates-Units Table, we propose a set of algorithms for the semantic type of quantity set computation, quantity optimizing, units of quantity inference, and final answer solving. The results of the experiments show a high correctness ratio, although there are some limitations on semantic knowledge.

Our approach is very useful for information processing area, particularly the combination of machine learning, domain ontologies, and a human-in-the-loop [30,31]. In future research, we will apply our approach to biomedical area. Additionally, we will extend the Predicates-Units Table to an ontology so that more semantic knowledge, such as *is-a* relation and *part-of* relation between units could be included. For instance, we will define *is-a* relation between units "area" and "totalarea", between units "area" and "landarea", and between units "area" and "waterarea". Furthermore, we will define *part-of* relation between units "totalarea" and "landarea", and between units "totalarea" and "waterarea". According to these relations, we are able to define equation "landarea + waterland = totalarea". The equation is very helpful to infer units of quantities. For query (*Russia*, *area*), quantities 1,699,580, 79,400, and 17,075,200 are fitting for the equation "landarea + waterarea = totalarea" rather than the ratio model. A useful conclusion is that units of quantities 1,699,580, 79,400, and 17,075,200 are the same. The conclusion can avoid inferring the unit quantities incorrectly. We anticipate that better experimental results will be obtained.

Acknowledgments: We thank all reviewers for their useful comments to improve the paper. This work was supported by the Fundamental Research Funds for the Central Universities (3132015043, 3132014036, 3132016348), National Natural Science Foundation of China (61203283), Liaoning Provincial Natural Science Foundation of China (2014025004), The Open Project Program of Artificial Intelligence Key Laboratory of Sichuan Province (2013RZJ02, 2012RYJ02).

Author Contributions: Yaqing Liu conceived the research subject and contributed to the critical suggestions of the paper, Lidong Wang carried out the experiments, Rong Chen drafted the paper, Yingjie Song and Yalin Cai contributed to the conception and critical suggestions of the paper. All authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, K.; Zhao, J.; He, S.; Zhang, Y. Question Answering over Knowledge Bases. *IEEE Intell. Syst.* **2015**, *30*, 26–35. [CrossRef]
2. Zheng, Y.; Jeon, B.; Xu, D.; Wu, Q.M.J.; Zhang, H. Image segmentation by generalized hierarchical fuzzy C-means algorithm. *J. Intell. Fuzzy Syst.* **2015**, *28*, 961–973.
3. Takamura, H.; Tsujii, J. Estimating Numerical Attributes by Bringing Together Fragmentary Clues. In Proceedings of the Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL, Denver, CO, USA, 31 May–5 June 2015; pp. 1305–1310.
4. Oren, E.; Delbru, R.; Catasta, M.; Cyganiak, R.; Stenzhorn, H.; Tummarello, G. Sindice.com: A document-oriented lookup index for open linked data. *Int. J. Metadata Semant. Ontol.* **2008**, *3*, 37–52. [CrossRef]
5. Davies, D.L.; Bouldin, D.W. A Cluster Separation Measure. *IEEE Trans. Pattern Anal. Mach. Intell.* **1979**, *2*, 224–227. [CrossRef]
6. Davidov, D.; Rappoport, A. Extraction and approximation of numerical attributes from the Web. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, 11–16 July 2010; pp. 1308–1317.
7. Moriceau, V. Generating Intelligent Numerical Answers in a Question-Answering System. In Proceedings of the Fourth International Natural Language Generation Conference, Sydney, Australia, 15–16 July 2006; pp. 103–110.
8. Maiya, A.S.; Visser, D.; Wan, A. Mining Measured Information from Text. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, 9–13 August 2015; pp. 899–902.

9. Banerjee, S.; Chakrabarti, S.; Ramakrishnan, G. Learning to rank for quantity consensus queries. In Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Boston, MA, USA, 19–23 July 2009; pp. 243–250.
10. Sarawagi, S.; Chakrabarti, S. Open-domain quantity queries on web tables: Annotation, response, and consensus models. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 711–720.
11. Mendes, A.C.; Coheur, L. When the answer comes into questions in question-answering: Survey and open issues. *Nat. Lang. Eng.* **2013**, *19*, 1–32. [CrossRef]
12. Heie, M.H.; Whittaker, E.W.D.; Furui, S. Question answering using statistical language modeling. *Comput. Speech Lang.* **2012**, *26*, 193–209. [CrossRef]
13. Moriceau, V. Numerical data integration for cooperative question-answering. In Proceedings of the Workshop KRAQ'06 on Knowledge and Reasoning for Language Processing, Trento, Italy, 3 April 2006; pp. 43–50.
14. Webber, B.; Gardent, C.; Bos, J. Position statement: Inference in question answering. In Proceedings of the LREC Workshop on Question Answering: Strategy and Resources, Las Palmas, Spain, 27 May–2 June 2002; pp. 19–25.
15. Nyberg, E.; Mitamura, T.; Carbonell, J.G.; Callan, J.P.; Collins-Thompson, K.; Czuba, K.; Duggan, M.; Hiyakumoto, L.; Hu, L.; Huang, Y.; *et al.* The JAVELIN question-answering system at TREC 2002. In Proceedings of the TREC'02, Gaithersburg, MD, USA, 19–22 November 2002.
16. Tellez-Valero, A.; Montes-ntes-Gomez, M.; Villasenor-Pineda, L.; Penas, A. Towards multi-stream question answering using answer validation. *Informatica* **2010**, *34*, 45–54.
17. Kwok, C.; Etzioni, O.; Weld, D.S. Scaling question answering to the web. *ACM Trans. Inf. Syst.* **2001**, *19*, 242–262. [CrossRef]
18. Brill, E.; Dumais, S.; Banko, M. An analysis of the AskMSR question-answering system. In Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, Philadelphia, PA, USA, 7–12 July 2002; pp. 257–264.
19. Bos, J.; Curran, J.R.; Guzzetti, E. The pronto QA system at TREC-2007: Harvesting hyponyms, using nominalisation patterns, and computing answer cardinality. In Proceedings of the 16th Text RETrieval Conference, Gaithersburg, MD, USA, 5–9 November 2007; pp. 726–732.
20. Nii, Y.; Kawata, K.; Yoshida, T.; Sakai, H.; Masuyama, S. Question answering system QUARK. In Proceedings of the NTCIR-4, Tokyo, Japan, 2–4 June 2004.
21. Pasca, M.; Harabagiu, S. The informative role of WordNet in open-domain question answering. In Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh, PA, USA, 2–7 June 2001; pp. 905–912.
22. Dalmas, T.; Webber, B. Answer comparison in automated question answering. *J. Appl. Log.* **2007**, *5*, 104–120. [CrossRef]
23. Harabagiu, S.; Hickl, A.; Lacatusu, F. Negation, contrast and contradiction in text processing. In Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06), Boston, MA, USA, 16–20 July 2006; pp. 755–762.
24. Marneffe, M.D.; Rafferty, A.; Manning, C.D. Finding contradictions in text. In Proceedings of the ACL-08: HLT, Columbus, OH, USA, 15–20 June 2008; pp. 1039–1047.
25. Gu, B.; Sheng, V.S.; Tay, K.Y.; Romano, W.; Li, S. Incremental Support Vector Learning for Ordinal Regression. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 1403–1416. [PubMed]
26. Gu, B.; Sheng, V.S.; Wang, Z.; Ho, D.; Osman, S.; Li, S. Incremental learning for ν-Support Vector Regression. *Neural Netw.* **2015**, *67*, 140–150. [CrossRef] [PubMed]
27. Li, X.; Huang, X.; Wu, L. Combined Multiple Classifiers Based on TBL Algorithm and Their Application in Question Classification. *J. Comput. Res. Dev.* **2008**, *45*, 535–541.
28. Sagara, T.; Hagiwara, M. Natural language neural network and its application to question-answering system. *Neurocomputing* **2014**, *142*, 201–208. [CrossRef]
29. Hu, B.S.; Wang, D.L.; Yu, G.; Ma, T. An Answer Extraction Algorithm Based on Syntax Structure Feature Parsing and Classification. *Chin. J. Comput.* **2008**, *31*, 662–676. [CrossRef]

30. Ma, T.; Zhou, J.; Tang, M.; Tian, Y.; Al-Dhelaan, A.; Al-Rodhaan, M.; Lee, S. Social network and tag sources based augmenting collaborative recommender system. *IEICE Trans. Inf. Syst.* **2015**, *E98-D*, 902–910. [CrossRef]
31. Girardi, D.; Küng, J.; Kleiser, R.; Sonnberger, M.; Csillag, D.; Trenkler, J.; Holzinger, A. Interactive knowledge discovery with the doctor-in-the-loop: A practical example of cerebral aneurysms research. *Brain Inform.* **2016**, 1–11. [CrossRef]