Information Sciences 355–356 (2016) 152–168

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A probabilistic ranking framework for web-based relational data imputation

Zhaoqiang Chen[a], Qun Chen[a,*], Jiajun Li[a], Zhanhuai Li[a], Lei Chen[b]

[a] School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an, 710072, PR China
[b] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, PR China

ARTICLE INFO

Article history:
Received 26 October 2015
Revised 4 February 2016
Accepted 16 March 2016
Available online 24 March 2016

Keywords:
Web-based relational data imputation
Missing attribute values
Probabilistic ranking

ABSTRACT

Due to richness of information on web, there is an increasing interest to search for missing attribute values in relational data on web. Web-based relational data imputation has to first extract multiple candidate values from web and then rank them by their matching probabilities. However, effective candidate ranking remains challenging because web documents are unstructured and popular search engines can only provide with relevant but not necessarily semantically matching information.

In this paper, we propose a novel probabilistic approach for ranking the web-retrieved candidate values. It can integrate various influence factors, e.g. snippet rank order, occurrence frequency, occurrence pattern, and keyword proximity, in a single framework by semantic reasoning. The proposed framework consists of snippet influence model and semantic matching model. The snippet influence model measures the influence of a snippet, and the semantic matching model measures the semantic similarity between a candidate value in a snippet and a missing relational value in a tuple. We also present effective probabilistic estimation solutions for both models. Finally, we empirically evaluate the performance of the proposed framework on real datasets. Our extensive experiments demonstrate that it outperforms the state-of-the-art techniques by considerable margins on imputation accuracy.

1. Introduction

A relational database is expected to present complete answers provided that a valid query is given. Unfortunately, relational data may be incomplete in practice [9], e.g., some attribute values are missing. Therefore, there is a need for relational data imputation, which fills in missing attribute values. Internal data imputation can be achieved by either statistical models [1,7,10–12,27] or similarity rules [28,32,35]. However, internal data imputation may fall short in the circumstance that a missing attribute value's reference context is unique (e.g., the author of a specific book in a book dataset without duplicate records). Therefore, there is an increasing interest to extract the missing attribute values from external data sources, e.g. web.

The existing solutions for web-based relational data imputation search for a missing attribute value in the relevant snippets retrieved by a search engine. Unlike relational data, web data are usually schemaless and unstructured. Optimized for

* Corresponding author. Tel.: +86 13299168988.
E-mail addresses: chenzhaoqiang@mail.nwpu.edu.cn (Z. Chen), chenbenben@nwpu.edu.cn (Q. Chen).

http://dx.doi.org/10.1016/j.ins.2016.03.036
0020-0255/

Table 1
Finding the publisher of "Mister Magnolia".

Tuple	ISBN	0006618790
	Title	Mister Magnolia
	Author	Quentin Blake
	Publication year	1981
	Publisher	?
Target value	HarperCollins Publishers	
Submitted query	"0006618790" "1981" "publisher"	
Search engine	Bing API on 2015-05-21	
# of total retrieved snippets	8	
Candidates & occurrences	"HarperCollins Publishers" = 1, "Picture Lions" = 7	

keyword queries, popular search engines can only provide with relevant but not necessarily semantically matching information. As a result, it is usually infeasible to directly screen out a matching relational value in the retrieved snippets. Instead, many relevant but non-matching values have to be extracted as candidates as well [29,31]. Therefore, web-based relational data imputation has to depend on effective ranking to pick a matching value from a set of extracted candidate values.

As in the web-based question answering systems [4,6], the ranking solutions for relational data imputation [16] can be built on the voting mechanism. In the basic form, the voting approach ranks a candidate value by its occurrence frequency in the snippets. In the extended form, it may also consider other features, e.g. occurrence pattern, snippet rank order and keyword proximity. It builds a ranker on each feature and computes a final score for a candidate by merging the ranking orders. The voting approach effectively exploits the redundancy characteristic of web data: a correct value usually has a high frequency of occurrences on web. However, in relational data imputation, the set of extracted candidates usually includes many frequent and relevant but non-matching values. Their presence may significantly compound the difficulty of accurate ranking. We illustrate the shortcoming of the voting approach by the example, as shown in Table 1. Suppose that the name of the publisher of the book titled "Mister Magnolia" is missing in a book dataset, and the keyword query is "0006618790 1981 publisher", in which "0006618790" and "1981" are the ISBN number and publication year of the book respectively. The returned snippets contain many occurrences of the value "Picture Lions", which is the name of the *book series*, while the matching value "HarperCollins Publishers" occurs much less frequently in these snippets. To some extent, this shortcoming can be alleviated by incorporating other features besides occurrence frequency in the voting framework. However, as we show in Section 5, the effectiveness of the voting approach is still limited by disjoint feature influence estimations.

We observe that the key challenge of effective ranking is to bridge the semantic gap between relational data and web data. Due to relational schema, the semantics of a missing attribute value in a relational tuple is well defined. However, it is much more challenging to reason about the semantics of a candidate value within a snippet because of the unstructured and noisy nature of web data. On the other hand, even though every factor so far considered in the literature can influence the matching probability of a candidate value, their disjoint influence estimations have not been systematically integrated yet. Therefore, there is a need for a unified ranking framework that can semantically reason about matching probability. This paper aims for such a framework. Our major contributions can be summarized as follows:

1. First, as far as we know, we are the first to study the ranking issue of web-based relational data imputation from a probabilistic and semantic perspective. The proposed approach seamlessly integrates various influence factors in a single framework by semantic reasoning. The resulting framework consists of two components, *snippet influence model* and *semantic matching model*, which measure the influence of a retrieved snippet and the matching probability of a candidate value within a snippet respectively.
2. Next, we present effective probabilistic estimation solutions for *snippet influence model* and *semantic matching model*. We propose a PageRank approach for snippet influence model that can effectively incorporate snippet rank order and content similarity in a single estimation process. Our solution for semantic matching model quantifies the matching probability by incrementally measuring the semantic similarity between a candidate value and a missing relational value.
3. Finally, we evaluate the performance of the proposed framework empirically on real datasets. Our extensive experiments demonstrate that it achieves considerably higher imputation accuracy than existing techniques.

The rest of this paper is organized as follows: Section 2 reviews related works. Section 3 sketches the web-based relational data imputation process. Section 4 describes the probabilistic ranking framework and the probabilistic solutions for snippet influence model and semantic matching model. Section 5 empirically evaluates the performance of the proposed framework. Finally, Section 6 concludes this paper.

2. Related work

The first prototype system for web-based relational data imputation using search engines was proposed in [16]. Li et al. focused on how to formulate effective keyword queries such that the missing values can be successfully retrieved. In a follow-up work [15], they also investigated the effective extraction techniques that can improve the recall level while at the same time including as few values as possible in the candidate set. In contrast, our work in this paper focused on a general

ranking framework that can accurately estimate the matching probability of extracted candidates. Its effectiveness does not depend on the goodness of keyword queries and candidate extraction techniques.

The web-based question answering systems [4,6,8,13,17,25] and statement truthfulness verification on web [14] also required to rank multiple extracted candidate entities/concepts. The question answering systems first employed a set of filters to yield reliable candidate answers and then ranked them in a voting approach by computing their scores on various features. As in question answering, the existing ranking techniques for statement truthfulness verification are also based on voting. We have presented and implemented a similar vote-based approach for web-based relational data imputation for comparative empirical study in this paper. Our experiments showed that the performance of the voting-based approach can still be limited by disjoint feature influence estimations.

There also exist some work on object-level web search [19,20] and object set expansion [30]. Similar to document retrieval, object retrieval ranked objects in term of their relevance and popularity in answering user queries. The work on set expansion instead studied how to expand an initial set of objects into a more comprehensive set. Both object-level web search and set expansion consider including multiple candidates in the final result set. They measure ambiguous relevance instead of the well-defined matching probability as required by relational data imputation. Therefore, their ranking approaches can not be applied to the task of relational data imputation.

From a broader perspective, semantic matching has also been studied in the contexts of ontology retrieval [2,3,26] and image retrieval [33,34]. For instance, [3] introduced a three-stage approach to semantically retrieve the most relevant ontology from a given repository in the circumstance that early requirements may be ambiguous, incomplete and/or inconsistent. [34] developed a semantic preserving distance metric learning method that encoded the feature similarity and semantic similarity in a unified feature space for image clustering task. In these works, the structures of ontology repository and feature space are well defined. In web-based relational data imputation, the snippets retrieved by search engines are instead unstructured. Our proposed framework is different from these ranking techniques in that it can effectively bridge the semantic gap between structured relational data and unstructured web data.

3. Background

In this section, we first formulate the task of candidate value ranking in Section 3.1, and then present a voting approach in Section 3.2.

3.1. Task formulation

Let R denote a relational schema, A denote an attribute in R, and T denote an instance of schema R. A tuple t in T is incomplete iff there exists missing attribute values in t. A tuple t's attribute value at A is said to be missing in T if it is equal to *null*. Automatic web-based relational data imputation is supposed to fill in the missing attribute values in T by retrieving and analyzing relevant web information. Abstracted in Fig. 1, it consists of the following three steps:

Step 1. Query construction: construct a set of keyword queries, $Q = \{q_1, q_2, \ldots, q_m\}$;
Step 2. Candidate extraction: retrieve a set of relevant web snippets by a search engine, $S = \{s_1, s_2, \ldots, s_n\}$ and extract a set of candidate values, $C = \{c_1, c_2, \ldots, c_l\}$, from the snippets in S;
Step 3. Candidate ranking: rank the candidate values in C by matching probability.

Unfortunately, every step in the above procedure poses challenges. In the step of query construction, a short query may retrieve many non-semantically-matching even irrelevant snippets. Their presence may significantly compound the difficulty of effective ranking. A long query may not work well either. It may retrieve only a short list of snippets, and thus decrease the success rate of a missing value being extracted. In the step of candidate extraction, strict techniques (e.g., pattern-based [22–24]) decrease the probability that a missing value be successfully extracted. In comparison, loose techniques (e.g., Name-Entity-Recognizer-based (NER-based) [18]) usually retrieve more false positive candidate values.

In this paper, we focus on the final step, candidate ranking. Note that whichever techniques used for query construction and candidate extraction, multiple candidate values have to be extracted and considered for matching. Therefore, effective candidate ranking is crucial to the accuracy of relational data imputation. It has been pointed out [16] that the techniques of using multiple queries and loose (rather than pattern-based) extraction approaches are effective in increasing the recall level of candidate extraction. Since these techniques usually generate more candidate values, they make the accuracy of relational data imputation even more dependent upon effective ranking.

For ease of presentation, we summarize the used notations in Table 2. We formulate the task of candidate value ranking as follows:

Definition 1 (Target attribute value)**.** Given an instance T of a relational schema R, suppose that a tuple $t \in T$ has a missing value at the attribute A. A target attribute value is the true attribute value at A of the tuple t, denoted by $t[A]$.

Definition 2 (Task of candidate value ranking)**.** Given a set of candidate values extracted from web, $C = \{c_1, c_2, \ldots, c_l\}$, considered for a missing attribute value $t[A]$, the task of candidate value ranking is to order the values in C by their probabilities of matching $t[A]$. Ideally, the target value $t[A]$ should be included in the set C and ranked first with the highest matching probability.

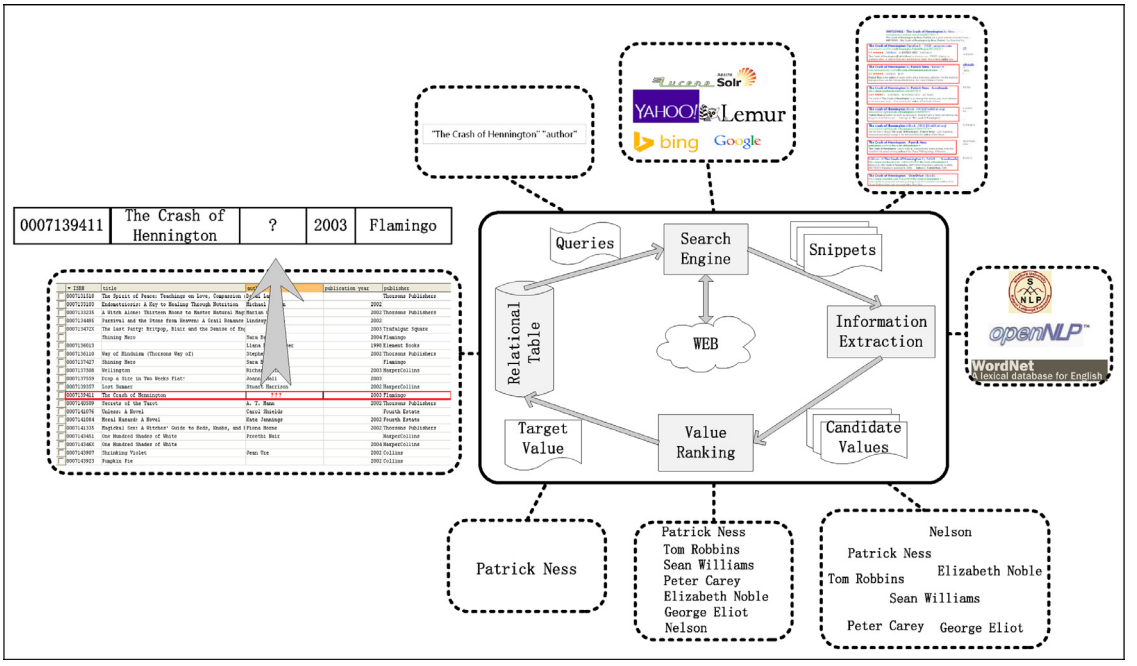


Fig. 1. Relational data imputation based on web (The surrounding dashed boxes are instances of the procedure.).

Table 2
Symbols and descriptions.

Symbol	Description
R	Relational schema
T	An instance of relational schema R
t	A tuple of T, where $t \in T$
A	An attribute in R
$t[A]$	The true value of attribute A (w.r.t. t)
S	A set of snippets (w.r.t. t and A)
s_m	An element of S, where $s_m \in S$
C	A set of candidate attribute values
c_i	An element of C, where $c_i \in C$
$\|D\|$	The cardinality of set D (i.e., number of elements of D)
K	The number of non-null attribute values of t
K_{s_m}	The number of non-null attribute values of t occurring in s_m
N	The total number of returned snippets in S
N_{c_i}	The number of snippets containing a candidate value c_i
S_{c_i}	The set of snippets containing c_i
ι	The average length of each snippet
ν	The number of candidate attribute values

3.2. A voting approach

Similar to web-based relational data imputation, the systems for web-based question answering and statement truthfulness verification [8,13,14,25] also require ranking components. They usually score candidate answers based on certain features and then compute a final rank order by merging the scores. Due to comprehensiveness of the features considered by the ranking technique proposed in [14] for statement truthfulness verification, we construct a voting-based ranking method based on these features.

Suppose that the tuple t with a missing attribute value at A has totally K non-null attribute values. Let N denote the total number of returned snippets in S and N_{c_i} denote the number of snippets containing a candidate value c_i. Given a snippet s_m containing c_i, let K_{s_m} denote the number of non-null attribute values of t occurring in s_m. The considered features include:

- Entity type. This feature considers whether the entity type of a candidate value matches the type of a missing target value.

- Snippet coverage. This feature measures the percentage of returned snippets that contain a candidate value. The snippet coverage of a candidate c_i is computed by:

$$SC(c_i) = \frac{N_{c_i}}{N}. \tag{1}$$

- Context relevance. This feature measures the relevance of a snippet to the relational context of a missing attribute value. The context relevance of a snippet s_m is estimated by $CR(s_m)=\frac{K_{s_m}}{K}$. The context relevance of a candidate c_i is measured by:

$$CR(c_i) = \frac{\sum_m CR(s_m)}{N_{c_i}} = \frac{\sum_m K_{s_m}}{K \cdot N_{c_i}}, \tag{2}$$

  in which s_m denotes a snippet containing c_i.
- Snippet rank. Snippet rank can also influence a candidate value's matching probability. The snippet rank of c_i is computed by aggregating the ranks of the snippets containing it:

$$SR(c_i) = \frac{\sum_m (1 - pos(s_m)/N)}{\sum_{1 \le n \le N} (1 - n/N)}, \tag{3}$$

  in which $pos(s_m)$ denotes the position of s_m in the rank list of returned snippets.
- Term distance. This feature measures the compactness of context term occurrences in a snippet. A snippet, where context terms occur closer to each other, is supposed to contain the target attribute value with a higher probability. Considering the smallest window of consecutive words that contains context terms in a snippet, we measure the feature of term distance by :

$$TD(c_i) = \frac{\sum_m ((1 - \frac{|L_m|}{|s_m|}) \cdot \frac{K_{s_m}}{K})}{N_{c_i}}, \tag{4}$$

  in which L_m denotes the smallest term window in s_m. In Eq. (4), we consider the length of the smallest term window as well as the number of context terms covered in a snippet. For term distance measurement, a snippet containing fewer context terms would automatically benefit from a smaller window size. We therefore penalize it by computing the portion of context terms it covers.
- Pattern matching. This feature measures the influence of occurrence pattern on a candidate's matching probability. A candidate, whose occurrence patterns in the snippets match the frequent occurrence patterns of similar values at the same attribute in T, is supposed to have a high probability of being the target value. Suppose that the top-k frequent occurrence patterns of the attribute values at A are $\{p_1, \ldots, p_k\}$ and each pattern p_j has an assigned weight of w_j. The feature influence of pattern matching is measured by:

$$PM(c_i) = \frac{\sum_j (w_j \cdot M_j)}{\sum_j w_j}, \tag{5}$$

  in which M_j denotes whether there exists an occurrence of c_i in returned snippets that matches a top-k pattern, $M_j = 1$ if c_i matches the pattern p_j and $M_j = 0$ otherwise.

Summarizing the features described above, the voting approach computes the ranking score of a candidate c_i using the following formula:

$$Score(c_i) = w_1 \cdot SC(c_i) + w_2 \cdot CR(c_i) + w_3 \cdot SR(c_i) + w_4 \cdot TD(c_i) + w_5 \cdot PM(c_i), \tag{6}$$

in which w_i(i = 1,...,5) denote the feature weights. In practice, feature weight assignment is usually optimized by training.

4. Probabilistic ranking framework

Given an incomplete tuple t with a missing value at the attribute A and a candidate value c_i, we denote the probability of c_i being the missing attribute value by $P(c_i|t)$. Since candidate values can only be extracted from snippets, $P(c_i|t)$ can be computed by applying the law of total probability as follows:

$$P(c_i|t) = \sum_{m=1}^{|S|} P(c_i, s_m|t). \tag{7}$$

Then, applying the Bayesian rule on the probability $P(c_i, s_m|t)$, we have

$$P(c_i|t) = \sum_{m=1}^{|S|} \frac{P(c_i, s_m) \cdot P(t|c_i, s_m)}{P(t)} = \sum_{m=1}^{|S|} \frac{P(s_m) \cdot P(c_i|s_m) \cdot P(t|c_i, s_m)}{P(t)}, \tag{8}$$

in which $P(s_m)$ denotes the retrieval probability of the snippet s_m, $P(c_i|s_m)$ denotes the probability of s_m containing c_i, and $P(t|c_i, s_m)$ denotes the probability of the semantic context of the candidate c_i in s_m matching that of the missing attribute value in t. Obviously, $P(c_i|s_m) = 1$ if the snippet s_m contains c_i, and otherwise $P(c_i|s_m) = 0$. Let S_{c_i} denote the set of snippets containing c_i. We have

$$P(c_i|t) = \sum_{m=1}^{|S_{c_i}|} \frac{P(s_m) \cdot P(t|c_i, s_m)}{P(t)}. \tag{9}$$

Note that the ranking framework only needs to measure the relative order of $P(c_i|t)$. We also observe that given a tuple t, the denominator of Eq. (9), $P(t)$, is a constant value. Therefore, the candidate values can be ranked by

$$P_r(c_i|t) = \sum_{m=1}^{|S_{c_i}|} \underbrace{P(s_m)}_{snippet\ influence\ model} \cdot \underbrace{P(t|c_i, s_m)}_{semantic\ matching\ model}. \tag{10}$$

In Eq. (10), the priori retrieval probability, $P(s_m)$, characterizes a snippet's influence on estimating a candidate value's matching probability. We quantify its estimation by *snippet influence model*. $P(t|c_i, s_m)$ represents the probability that given a snippet s_m and a candidate value occurring in s_m, the semantic context of c_i in s_m matches that of the target attribute value in t. We quantify its estimation by *semantic matching model*.

4.1. Snippet influence model

A snippet's position in the rank list indicates its relevance to a keyword query, thus to the semantic context of the incomplete tuple t. A snippet ranked higher by a search engine should be considered to have higher influence. We also observe that the snippets retrieved by a keyword query usually have similar contents. Due to the redundancy characteristic of web data, the information in a snippet can be supposed to be more reliable if it is supported by more sources. Correspondingly, a snippet with a higher degree of content similarity with regard to other snippets should be considered to have higher influence. In this subsection, we propose a PageRank [21] approach for measuring snippet influence that can integrate these two factors in a single model.

PageRank is a recursive graph-based random walk procedure. Imagine that an alien lives in a graph and he can reach any vertex along edges or by directly landing at it in a random manner. The PageRank procedure computes the probability that every vertex is reached by the alien. We construct a snippet graph G, whose vertices V and edges E represent the snippets and their content similarity respectively. The graph is undirected and complete since there exists an edge between every pair of snippets. The common solutions to measure content similarity between snippets are based on *Jaccard index*, a.k.a. *Jaccard similarity coefficient*, or *Cosine similarity*. We have evaluated both solutions and found that they achieved similar performance. Since Jaccard index is easier to compute, we use it to measure content similarity between snippets. Let $TS(s_m)$ represent the token set of the snippet s_m. Given two snippets s_m and s_n, their content similarity is measured by

$$CS(s_m, s_n) = \frac{|TS(s_m) \cap TS(s_n)|}{|TS(s_m) \cup TS(s_n)|}. \tag{11}$$

Correspondingly, the edge weight between s_m and s_n, $W(s_m, s_n)$, is set to be $CS(s_m, s_n)$.

In the PageRank model, a snippet's retrieval probability corresponds to its probability of being reached by an alien randomly walking on the snippet graph G. We denote the proposed computational algorithm by SSRank, which stands for Similarity-based Snippet Re-rank method. The recursive computation of retrieval probability on G is specified by

$$SSRank_{k+1}(s_m) = \lambda \cdot J_{s_m} + (1 - \lambda) \cdot \sum_{s_n \in V \& s_n \neq s_m} \frac{SSRank_k(s_n) \cdot W(s_m, s_n)}{\sum_{s_j \in V \& s_j \neq s_n} W(s_n, s_j)}, \tag{12}$$

in which λ represents the dampening factor, the first part $(\lambda \cdot J_{s_m})$ represents the probability of directly landing at s_m by random selection, and the second part represents the probability of indirectly reaching s_m from other vertices along the edges.

In Eq. (12), the probability of an alien located at s_n walking along the edge to s_m is set to be proportional to the weight of the edge, $W(s_m, s_n)$, or the content similarity between s_m and s_n. The probability of random jump to a snippet, J_{s_m},

depends on its ranking position in the retrieved snippet list. The higher rank order a snippet s_m has, the higher value its corresponding J_{s_m} has. We set the value of J_{s_m} by

$$J_{s_m} = \frac{PosWeight(s_m)}{\sum_{n=1}^{|S|} PosWeight(s_n)}, \tag{13}$$

where $PosWeight(s_n)$ denotes the position weight of the snippet s_n in the retrieved list. The value of $PosWeight(s_n)$ is specified by

$$PosWeight(s_n) = \frac{1}{\log_2(1 + Pos(s_n))}, \tag{14}$$

in which $Pos(s_n)$ represents the rank order of s_n in the retrieved list. It can be observed that the value of $PosWeight(s_n)$ decreases with the rank order. For instance, if s_n is ranked first by a search engine, it has the highest rank order of 1. With $Pos(s_n) = 1$, we have $PosWeight(s_n) = \frac{1}{\log_2(1+1)} = 1$.

In Eq. (12), we set the value of λ to 0.15 and the initial values of $SSRank_0(s_m)$ to $\frac{1}{|V|}$. The values of $SSRank_{k+1}(s_m)$ are recursively computed until they converge. The complete procedure of Snippet Influence Model is sketched in Procedure 1.

Procedure 1 Similarity-based Snippet Re-rank method(SSRank).

Input: G is the snippet graph;
Output: A vector $SSRank_{k+1}$ stores each snippet's score (i.e. $P(s_m)$);

procedure SSRank(G)
  $(V, E) \leftarrow G$ ▷ Split graph G into snippets and links.
  $SSRank_k \leftarrow$ a vector of length $|V|$ ▷ The current SSRank estimate.
  $SSRank_{k+1} \leftarrow$ a vector of length $|V|$ ▷ The resulting better SSRank estimate.
  $J \leftarrow$ a vector of length $|V|$
  for all $s_m \in V$ **do**
    $SSRank_k(s_m) \leftarrow \frac{1}{|V|}$ ▷ Initially, each snippet is equally likely to be a start point.
  end for
  for all $s_m \in V$ **do**
    $J_{s_m} \leftarrow$ The probability of random jump to a snippet. ▷ Eqs. (13) and (14)
  end for
  while $SSRank_{k+1}$ has not converged **do**
    for all $s_m \in V$ **do**
      $SSRank_{k+1}(s_m) \leftarrow \lambda \cdot J_{s_m}$
    end for
    for all snippet $s_n \in V$ **do**
      $AS \leftarrow$ the set of snippets such that $s_m \in V$ and $(s_m, s_n) \in E$ and $W(s_m, s_n) \neq 0$
      $WS \leftarrow \sum W(s_m, s_n)$, where $s_m \in V$ and $(s_m, s_n) \in E$ and $W(s_m, s_n) \neq 0$
      if $WS > 0$ **then**
        for all snippet $s_m \in AS$ **do**
          $SSRank_{k+1}(s_m) \leftarrow SSRank_{k+1}(s_m) + \frac{(1-\lambda) \cdot SSRank_k(s_n) \cdot W(s_m, s_n)}{WS}$
        end for
      else
        for all snippet $s_m \in V$ **do**
          $SSRank_{k+1}(s_m) \leftarrow SSRank_{k+1}(s_m) + \frac{(1-\lambda) \cdot SSRank_k(s_n)}{|V|}$
        end for
      end if
      $SSRank_k \leftarrow SSRank_{k+1}$
    end for
  end while
  return $SSRank_{k+1}$
end procedure

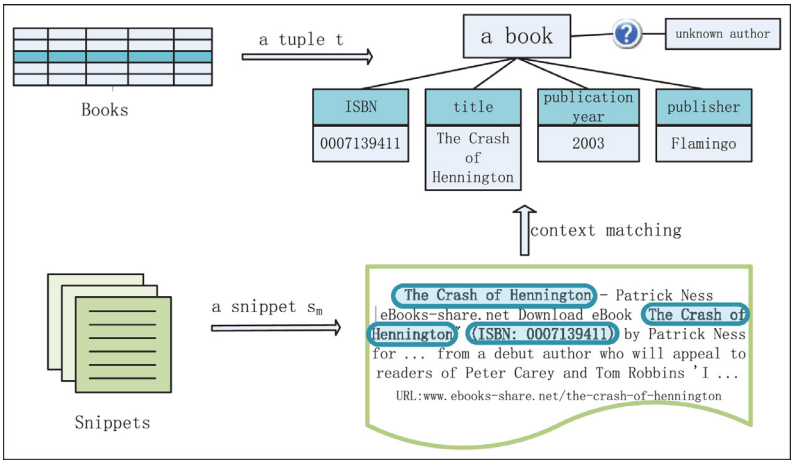


Fig. 2. Context matching.

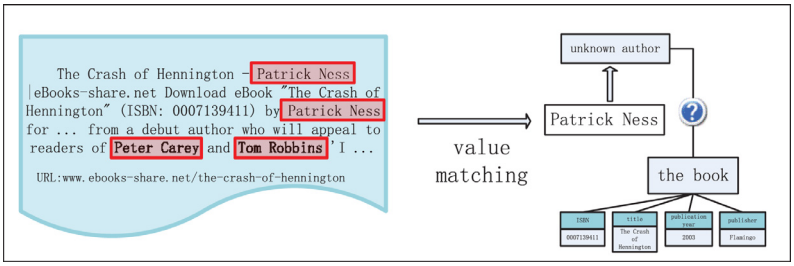


Fig. 3. Value matching.

4.2. Semantic matching model

The semantic matching model quantifies $P(t|c_i, s_m)$, the probability that the semantic context of a candidate value c_i in a snippet s_m matches that of the target attribute value missing in the tuple t. We represent the estimation of $P(t|c_i, s_m)$ by

$$P(t|c_i, s_m) = \frac{P(t, c_i, s_m)}{P(s_m) \cdot P(c_i|s_m)} = \frac{P(t, s_m)}{P(s_m)} \cdot \frac{P(t, c_i, s_m)}{P(t, s_m)} = \underbrace{P(t|s_m)}_{\text{context matching probability}} \cdot \underbrace{P(c_i|t, s_m)}_{\text{value matching probability}}, \tag{15}$$

where $P(c_i|s_m) = 1$ since the snippet s_m contains the candidate value c_i. In Eq. (15), $P(t|s_m)$ represents the probability that the semantic context of s_m matches that of t *without regard to* the missing attribute value in t. $P(c_i|t, s_m)$ represents the probability that c_i is the missing attribute value, provided that the semantic context of s_m matches that of t. By Eq. (15), $P(t|c_i, s_m)$ can be incrementally estimated by measuring two semantic components, $P(t|s_m)$ and $P(c_i|t, s_m)$.

We illustrate the semantics of context matching probability and value matching probability by the example of a book dataset. Suppose that t specifies a book, its author is the missing target attribute value, and its other non-null attribute values serve as context keywords. The semantic context of t, as shown in Fig. 2, defines the identity of the book, and the semantic pattern of the missing value, as shown in Fig. 3, specifies that it is the value at the attribute *author* of the book. Then, the context matching probability, $P(t|s_m)$, corresponds to the probability that the information in s_m is about the book as specified by t. The value matching probability, $P(c_i|t, s_m)$, corresponds to the probability that the candidate value c_i in s_m is the author of the book, under the assumption that s_m is indeed about the book.

4.2.1. Estimation of $P(t|s_m)$

The probability of $P(t|s_m)$ quantifies the context similarity between the snippet s_m and the tuple t. We consider its two orthogonal influence factors, *term coverage* and *term compactness*. We estimate $P(t|s_m)$ by

$$P(t|s_m) = f_{cove}(s_m) \cdot f_{comp}(s_m), \tag{16}$$

in which $f_{cove}(s_m)$ and $f_{comp}(s_m)$ denote the factors of term coverage and term compactness respectively. Term coverage measures the percentage of the non-null attribute values of t occurring in s_m. More context terms a snippet contains, more

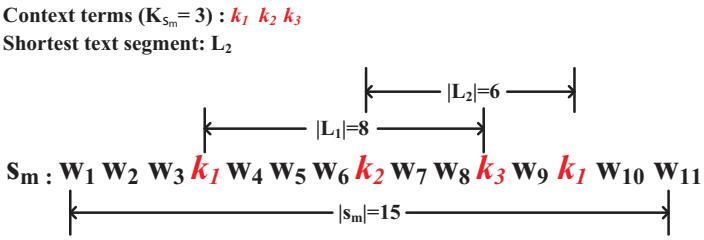


Fig. 4. A toy example of calculating term compactness. w_i represents a word in the text of snippet s_m.

probably its semantic context matches that of the tuple t. Term compactness measures the occurrence distance between context terms. More closer they occur in a snippet, more probably the snippet's context matches that of t.

■ *Term coverage*

As shown in Eq. (2) in Section 3, term coverage can be simply measured by

$$f_{cove}(s_m) = \frac{K_{s_m}}{K}, \tag{17}$$

where K denotes the total number of non-null attribute values of t and K_{s_m} denotes the number of non-null attribute values of t occurring in s_m.

Eq. (17) assumes that different attribute values are equally important in determining the semantic context of s_m. In practice, however, it is more likely that their importance are unequal. For instance, consider a tuple in a book dataset that contains the *ISBN, title, author* and *publisher* attribute values. It can be observed that a book's *ISBN* number and *title* are usually distinct, while its *author* and *publisher* values may be shared by many books. Therefore, the *ISBN* and *title* attribute values are more effective than the *author* and *publisher* values in determining the book identity. By weighting different attribute values, we can measure term coverage of s_m by an improved metric as follows:

$$f'_{cove}(s_m) = \frac{\sum_{A\in\mathbb{A}_{s_m}} AttrWeight(A)}{\sum_{A\in\mathbb{A}} AttrWeight(A)}, \tag{18}$$

where $\mathbb{A}$ represents the set of attributes whose values are non-null in t, $\mathbb{A}_{s_m}$ represents the set of attributes whose values occur in s_m and $AttrWeight(A)$ denotes the importance weight of the attribute A. We measure the importance weight of an attribute, A, by its distinctness in the tuples as follows:

$$AttrWeight(A) = \frac{|\{t[A]|t \in T\}|}{|T|}, \tag{19}$$

where the denominator ($|T|$) represents the total number of tuples in T, and the numerator denotes the number of different values of the attribute A occurring in T. If A is a primary key of a table, its importance weight would be maximal with the value of 1.

■ *Term compactness*

We consider the smallest window of consecutive words in s_m that contains all the context terms occurring in s_m. Note that each term has to occur only once in the specified text segment. By *the smallest window*, we mean that the text segment has the minimal number of words. Intuitively, a shorter text segment implies that the context terms of t occur closer to each other. Correspondingly, s_m's semantic context matches that of t with a higher probability.

With the help of an exponential function e^x and a damping factor μ where $0 < \mu \leq 1$, we measure term compactness by

$$f_{comp}(s_m) = e^{\mu\cdot(-\frac{|L_m|}{K_{s_m}|s_m|})}, \tag{20}$$

where $|L_m|$ denotes the length of the shortest text segment L_m and K_{s_m} denotes the number of context terms occurring in s_m. In Eq. (20), $\frac{|L_m|}{K_{s_m}}$ measures the average compactness of terms in s_m. Normalized by the length of a snippet, $|s_m|$, the average compactness captures the relative proximity between context terms in a snippet. A toy example of calculating term compactness has been shown in Fig. 4.

4.2.2. *Estimation of* P(c$_i$|t, s$_m$)

$P(c_i|t, s_m)$ quantifies the probability that a candidate value c_i in s_m *is* the missing attribute value in t, provided that the semantic context of s_m matches that of t. We consider two orthogonal factors that influence the probability of $P(c_i|t, s_m)$, *Value Distance* and *Pattern Matching*. *Value distance* measures a candidate value's proximity to the context terms of t. *Pattern matching* instead measures the probability from the perspective of occurrence pattern. We estimate $P(c_i|t, s_m)$ by

$$P(c_i|t, s_m) = f_d(s_m, c_i) \cdot f_p(s_m, c_i), \tag{21}$$

where the functions f_d and f_p denote the influence factors of *value distance* and *pattern matching* respectively.

■ *Value distance*

Consider the smallest window of consecutive words in the snippet s_m, which contains all the context terms occurring in s_m and the candidate value c_i. We denote it by $Win(c_i, t)$.

As in the measurement of term compactness, the smallest window means that the corresponding text segment has the minimal number of words and each context term has to occur only once in the window. Within $Win(c_i, t)$, we denote the term window beginning with the first occurrence of a context term and ending at the last occurrence of a context term by $Win(t)$. We measure the value distance of c_i by

$$f_d(s_m, c_i) = \mu + (1 - \mu) \cdot \frac{|Win(t)|}{|Win(c_i, t)|}, \tag{22}$$

in which μ is a dampening factor and $0 < \mu < 1$. According to Eq. (22), in the case that the candidate value c_i occurs between two context terms in $Win(c_i, t)$, the two windows, $Win(c_i, t)$ and $Win(t)$, are equal. As a result, $f_d(s_m, c_i)$ achieves the maximal value of 1. In the case that $Win(c_i, t)$ and $Win(t)$ are not equal, the value of $f_d(s_m, c_i)$ depends on the proximity of c_i to $Win(t)$.

■ *Pattern matching*

In the voting approach presented in Section 3, the factor of pattern matching is evaluated in a batch mode by considering all the snippets containing a candidate value. Our framework instead uses it to reason about the matching probability of a candidate value within a single snippet. With the knowledge of the top-k occurrence patterns, we measure the influence of pattern matching by

$$f_p(s_m, c_i) = \frac{\sum_j (w_j \cdot M_j) + w_0}{\sum_j w_j + w_0}, \tag{23}$$

where $M_j = 1$ if c_i has a top-k occurrence pattern of p_j in s_m, otherwise $M_j = 0$; w_j denotes the weight of the occurrence pattern p_j.

Note that the metric presented in Eq. (23) has a smooth factor of w_0. Based on the *Open-World Assumption*, the learned top-k patterns are only part of all possible occurrence patterns of the target attribute value. Without the smooth factor, the value of $f_p(s_m, c_i)$ would be estimated to 0 if c_i has no top-k occurrence pattern in s_m. In practice, we set the value of w_0 to 0.1.

4.2.3. Put all together

The complete estimation procedure of Semantic Matching Model is sketched in Procedure 2.

Procedure 2 Semantic match.

Input: a tuple t, a snippet s_m, a candidate value c_i that appear in s_m.
Output: an estimated value of $P(t|c_i, s_m)$.

```
1: procedure SemanticMatch(t,s_m,c_i)
2:     f_cove ← calculate Term Coverage (Eq. (18))
3:     f_comp ← calculate Term Compactness (Eq. (20))
4:     P(t|s_m) ← f_cove · f_comp                    ▷ context matching probability
5:     f_d ← calculate Value Distance (Eq. (22))
6:     f_p ← calculate Pattern Matching (Eq. (23))
7:     P(c_i|t, s_m) ← f_d · f_p                     ▷ value matching probability
8:     P(t|c_i, s_m) ← P(t|s_m) · P(c_i|t, s_m)
9:     return P(t|c_i, s_m)
10: end procedure
```

4.3. Ranking procedure and complexity analysis

The whole ranking procedure has been sketched in Procedure 3 . Let N denote the number of snippets and ι denote the average length of each snippet. For the "SSRank" procedure, the first step constructs a complete graph. It involves calculating content similarity between every pair of snippets. Its time complexity is $O(\iota \cdot N^2)$. Assume that the random walk procedure needs τ iterations to converge. The time complexity of random walk can be represented by $O(\tau \cdot N^2)$. Without loss of generality, the time complexity of the "SSRank" procedure can be represented by $O((\iota + \tau) \cdot N^2)$.

For the "SemanticMatch" procedure, suppose that there are ρ types of frequent patterns. We also assume that the occurrence frequencies of context terms and candidate values in each snippet is small, thus treated as a constant in time complexity analysis. Given a candidate, detecting a frequent pattern in a snippet requires to scan the snippet content once. Calculating term coverage, term compactness and value distance also requires to scan the snippets. Without loss of generality, the time complexity of the "SemanticMatch" procedure can be represented by $O(\rho \cdot \iota)$.

Procedure 3 A probabilistic ranking method.

Input: a tuple t, a set of snippets S, a set of candidate values C
Output: an ordered list of C

procedure RANKING(t,S,C)
  $RankList \leftarrow$ a list of length $|C|$ ▷ Each element in $RankList$ is a $< Key, Value >$ pair, whose Key represents a candidate and $Value$ represents the candidate's score that is initialized to 0.
  $G \leftarrow S$ ▷ Construct a graph according to a snippet set's information.
  $SSRank_{k+1} \leftarrow$ SSRANK(G) ▷ Snippet Influence. Details of $SSRank(G)$ refer to Procedure 1 in Subsection 4.1.
  for all snippet $s_m \in S$ **do**
    $P_1 \leftarrow SSRank_{k+1}(s_m)$
    for all candidate $c_i \in C_{s_m}$ **do** ▷ C_{s_m} represents a set of candidates that appear in s_m, $C_{s_m} \subseteq C$
      $P_2 \leftarrow$ SEMANTICMATCH(t,s_m,c_i) ▷ Semantic Matching. Details of $SemanticMatch(t,s_m,c_i)$ refer to Procedure 2 in Subsection 4.2.
      $RankList(c_i).Value \leftarrow RankList(c_i).Value + P_1 \cdot P_2$
    end for
  end for
  sort the $RankList$ in descending order by $RankList.value$.
  $C \leftarrow RankList.Key$
  return C
end procedure

Table 3
Samples of *BX* dataset.

ISBN	Title	Author	Publication year	Publisher
⋮	⋮	⋮	⋮	⋮
0001010565	Mog's Christmas	Judith Kerr	1992	Collins
0001047213	The Fighting Man	Gerald Seymour	1993	HarperCollins
0002243016	Desperadoes	Joseph O'Connor	1994	Flamingo
⋮	⋮	⋮	⋮	⋮

Therefore, we have the following theorem:

Theorem 1. *Assume that the occurrence frequencies of context terms and candidate values in the retrieved snippets are small, thus treated as a constant in time complexity analysis. The time complexity of the whole ranking procedure is* $O((\iota + \tau) \cdot N^2 + \rho \cdot \iota \cdot \nu \cdot N)$, *in which* ν *represents the number of extracted candidate values.*

5. Experimental evaluation

This section empirically evaluates the performance of the proposed ranking framework by comparative study. It is organized as follows: Section 5.1 presents two real test datasets. Section 5.2 presents the experimental setup, the alternative ranking approaches and evaluation metrics. Section 5.3 presents the comparative experimental results. Section 5.4 evaluates how the performance of different ranking approaches vary with the numbers of analyzed snippets. Section 5.5 evaluates how their performance vary provided with different query patterns. Finally, Section 5.6 evaluates how their performance vary with a stricter candidate extraction technique.

5.1. Datasets

The two test relational datasets are:

- **Book-Crossing** [36] **(BX).** This dataset was collected by Cai-Nicolas Ziegler in a 4-week crawl (August / September 2004) from the Book-Crossing community with kind permission from Ron Hornbaker, Chief Technology Officer of Humankind Systems. It contains 278,858 users (anonymous but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books[1]. A sample of the *BX* dataset is presented in Table 3.
- **World Information (world).** The *world* database, provided by Statistics Finland[2], is pre-installed in MySQL (Version 5.5). Our experiments use the *country* table that stores information about countries of the world. The table contains 239 countries and has 15 attributes, e.g. *country name*, *continent*, *region*, *capital* and *surface area*. A sample of the *country* dataset is also presented in Table 4.

[1] http://www2.informatik.uni-freiburg.de/~cziegler/BX/
[2] http://www.stat.fi/worldinfigures

Table 4
Samples of *world* dataset.

Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	⋯	Capital
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋯	⋮
AGO	Angola	Africa	Central Africa	1246700.00	1975	12878000	⋯	Luanda
AIA	Anguilla	North America	Caribbean	96.00	Null	8000	⋯	The Valley
ALB	Albania	Europe	Southern Europe	28748.00	1912	3401200	⋯	Tirana
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋯	⋮

5.2. Experimental setup

We compare the proposed probabilistic ranking framework, denoted by *Prob*, with four other ranking methods, which include:

- Frequency-based Ranking method (abbr. Freq). The work on web-based question answering [8,29] pointed out that a right answer usually has high occurrence frequency in a redundant large corpus. This observation is also largely true in the application of web-based relational data imputation. Therefore, we use the simple frequency-based ranking method as the baseline.
- Confidence-based Ranking method (abbr. Conf). Proposed in [15], it ranks a candidate by considering three parameters, document confidence, candidate frequency and the distance between candidate and context terms. Specifically, it measures the matching probability of a candidate β_c by a heuristic formula:

$$P(\beta_c|\alpha) = \frac{\sum_{d\in Docs} conf(d)\cdot s(\beta_c, d)}{\sum_{d\in Docs} conf(d)}, \tag{24}$$

  where $conf(d)$ is the confidence of document d, and $s(\beta_c, d)$ is the local score of β_c in d. $s(\beta_c, d)$ is estimated by:

$$s(\beta_c, d) = w\cdot\frac{freq(\beta_c, d)}{N} + (1-w)\cdot\sum_{1\le i\le freq}\frac{|d| - dist_i(\beta_c, \alpha)}{freq(\beta_c, d)\cdot|d|}, \tag{25}$$

  where $|d|$ is the length of document d, $freq$ is the frequency of β_c in d, $dist_i$ is the distance between the ith mention of β_c and the query entity α in d, N is a normalization factor and w is a scaling factor.
- Vote-based Ranking method (abbr. Vote). We have also implemented a voting approach for web-based relational data imputation as presented in Section 3 for comparative purpose. It was structured like the ranking framework originally proposed for web-based statement truthfulness verification [14].
- Context-aware Entity Ranking method (abbr. CER). Proposed in [5], it first constructs an entity graph based on the relationship between extracted entities and then ranks the candidate entities through random walk in the graph. The CER approach aims to take advantage of the relationship between web entities (as represented by graph model) to enable effective ranking.

The imputation accuracy of the ranking methods are compared on two metrics, **# Top-k** and **MRR**:

- **# Top-k.** It measures the number of missing attribute values that are ranked in the top-k positions in the ordered list. The metric of top-1 is the most important indicator of ranking accuracy because relational data imputation usually requires a single value for a missing one. In our experiments, we use the metrics of top-1 and top-3.
- **MRR.** Standing for *Mean Reciprocal Rank*, it is computed by

$$MRR = \frac{1}{N}\sum_t \frac{1}{rank_{t[A]}}, \tag{26}$$

  where N is the number of test tuples with missing attribute values and $rank_{t[A]}$ represents the position of a target attribute value in the candidate ranking list. Complementary to the metric of Top-k, the metric of MRR measures the overall ranking accuracy by considering both the top-ranked target values and those not ranked in the top positions.

In our experiments, we use *Bing Search API*[3], provided by Microsoft, to retrieve relevant documents from web. The performance of various ranking approaches are evaluated using both the loose and strict candidate extraction techniques. The loose technique uses Name Entity Recognizer [18] to extract candidate attribute values from unstructured text. We used the value extracting module of an open source Question&Answering system named Ephyra[4]. It integrates a series of *Natural Language Processing* technologies, which include (Conditional Random Field) CRF-NER[5] from Stanford, openNLP[6] from

[3] https://datamarket.azure.com/dataset/bing/search
[4] http://www.ephyra.info
[5] http://nlp.stanford.edu/software/CRF-NER.shtml
[6] http://opennlp.apache.org

Table 5
The patterns of submitted queries of two datasets.

Datasets	Attributes	Query pattern
BX	ISBN	t[Title]+t[Publication Year]+"ISBN"
	Title	t[ISBN]
	Author	t[Title]
	Publication year	t[ISBN]
	Publisher	t[ISBN]+t[Publication Year]+"publisher"
world	Continent	t[Name]+"continent"
	Region	t[Name]+"region"
	SurfaceArea	t[Name]+"surface area square km"
	IndepYear	t[Name]+"independence"
	Capital	t[Name]+"capital"

Apache, WordNet[7] from Princeton and Snowball[8]. In the strict case, we filter the candidates extracted by NER by the learned occurrence patterns. To be included in the candidate set, a value should have at least a referred occurrence pattern in the retrieved snippets.

5.3. A comparative study

In the *BX* dataset, our experiments impute the attribute values at *author*, *title*, *isbn*, *publication year* and *publisher*. In the *world* dataset, our experiments impute the attribute values at *region*, *surface area*, *continent*, *capital* and *independent year*. The number of missing values at each attribute and its corresponding keyword query pattern have been presented in Table 5. For the *world* dataset, the average number of returned snippets per keyword query is 282, while the number is only 61 for the *BX* dataset. We, therefore, set the number of analyzed snippets to 300. Our experiments show that setting it to be more than 300 does not result in improved imputation recall and accuracy.

The detailed evaluation results on the *BX* dataset are presented in Table 6. The values at the top-k columns of the table represent the percentage of target attribute values ranked in the top-k positions. It can be observed that *Prob* achieves the overall best ranking accuracy among the tested methods. The *Freq* method may not perform well in the circumstance that there exist some highly frequent non-matching values in the candidate set. In the example shown in the introduction, *Freq* fails to rank the matching value "HarperCollins Publisher" before another candidate "Picture Lions". By considering other relevant features besides occurrence frequency, *Vote* achieves an overall better accuracy than *Freq*. However, it remains very challenging to optimally assign the influence weights of different features. In *Vote*, with the learned weight assignment, a candidate's score on some feature may dominate its scores on other features. Ranking biased towards this feature may result in poor accuracy. Again, in the example shown in the introduction, the candidate "Picture Lions" has a very high occurrence frequency in returned snippets. Its final score in *Vote* is dominated by its score on occurrence frequency. As a result, it is still ranked before the matching value "HarperCollins Publisher", which has a low frequency, even though its scores on other features are low.

It can be observed that *Prob* consistently outperforms *Conf* on ranking accuracy. Even though the *Conf* approach integrates the local score of a candidate within a snippet and confidence measurement of the snippet in a single model, our experiments show that its performance is still limited by linear combination of frequency and distance in the computation of local score. It is also worthy to point out that *Prob* achieves an overall better performance than *CER*, even though *CER* performs better on the *Title* attribute. The performance advantage of *Prob* over *Conf* and *CER* clearly demonstrates the effectiveness of semantic reasoning.

The detailed evaluation results on the *world* dataset are presented in Table 7. Similar to what are observed on the *BX* dataset, *Prob* achieves the overall best ranking accuracy among all the tested approaches. In the case of *surface area*, the candidates should be of the number type. Besides the area number, other trivial numbers, e.g. 1, 2 and 2015, may be extracted as candidates as well. Since these numbers usually have high occurrence frequency in retrieved snippets, *Freq* performs very poorly in ranking the candidates for *surface area*.

5.4. Varying the number of analyzed snippets

We track the performance of different ranking approaches as the upper-bound number of analyzed snippets increases from 1 to 300. The detailed evaluation results on the *author* and *title* attributes of *BX* and the *region* and *surface area* of *world* are presented in Fig. 5. In the figure, the X-axis values represent the number of snippets and the Y-axis values represent the percentage of top ranked target values. As expected, the imputation success rate initially increases dramatically and then

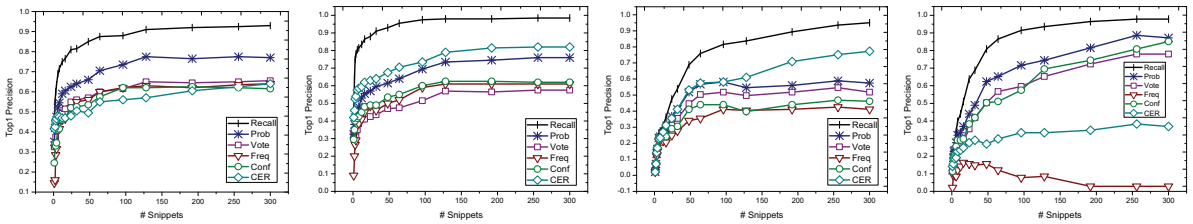
[7] http://wordnet.princeton.edu
[8] http://snowball.tartarus.org

Table 6
Ranking results of BX dataset (Maximum # of Snippets = 300).

			Loose extraction (e.g. type-based)				Strict extraction (e.g. pattern-based)			
	# Testing tuples	Ranking method	# Recall	Top1	Top3	MRR	# Recall	Top1	Top3	MRR
Overview	996	Vote	884	0.6912	0.9038	0.8030	704	0.8466	0.9872	0.9131
		Conf		0.7036	0.8993	0.8097		0.8594	0.9844	0.9220
		CER		0.7432	0.9344	0.8403		0.8622	0.9858	0.9231
		Freq		0.6674	0.8676	0.7785		0.8267	0.9801	0.9025
		Prob		**0.7862**	**0.9434**	**0.8688**		**0.8778**	**0.9915**	**0.9326**
ISBN	197	Vote	171	0.8538	0.9649	0.9102	169	0.8639	0.9704	0.9175
		Conf		0.8889	0.9766	0.9354		0.8935	0.9822	0.9400
		CER		0.8480	0.9649	0.9112		0.8580	0.9763	0.9202
		Freq		0.8304	0.9591	0.9011		0.8402	0.9763	0.9108
		Prob		**0.8947**	**0.9883**	**0.9409**		**0.9053**	**0.9941**	**0.9482**
Title	200	Vote	197	0.5838	0.8274	0.7194	171	0.8070	1	0.8938
		Conf		0.6294	0.8782	0.7614		0.8304	0.9942	0.9089
		CER		0.8325	0.9695	0.8982		0.8947	1	0.9464
		Freq		0.6193	0.9036	0.7635		0.8070	0.9942	0.8991
		Prob		**0.7716**	**0.9239**	**0.8571**		**0.9123**	**1**	**0.9542**
Author	200	Vote	186	0.7043	0.8763	0.8001	171	0.9123	0.9942	0.9466
		Conf		0.6613	0.8548	0.7683		0.8830	0.9883	0.9332
		CER		0.6882	0.8660	0.7860		0.8655	0.9766	0.9172
		Freq		0.6882	0.8387	0.7703		0.8830	0.9766	0.9301
		Prob		**0.8280**	**0.9032**	**0.8758**		**0.9181**	**0.9942**	**0.9515**
Publisher year	200	Vote	181	0.7348	0.9558	0.8461	64	0.8906	1	0.9427
		Conf		0.7901	0.9669	0.8789		0.9531	1	0.9766
		CER		0.7403	0.9724	0.8520		0.9531	1	0.9766
		Freq		0.7238	0.8950	0.8226		0.9219	1	0.9584
		Prob		**0.7956**	**0.9779**	**0.8860**		**0.9531**	**1**	**0.974**
Publisher	199	Vote	149	0.5772	0.9060	0.7418	129	0.7674	0.9767	0.8740
		Conf		0.5369	0.8121	0.6969		0.7752	0.9612	0.8740
		CER		0.5772	0.8926	0.7358		0.7752	0.9845	0.8773
		Freq		0.4497	0.7181	0.6143		0.7132	0.9612	0.8319
		Prob		**0.6174**	**0.9262**	**0.7717**		**0.7054**	**0.9690**	**0.8379**

Table 7
Ranking results of world dataset (Maximum # of Snippets = 300).

			Loose extraction (e.g. type-based)				Strict extraction (e.g. pattern-based)			
	# Testing tuples	Ranking method	# Recall	Top1	Top3	MRR	# Recall	Top1	Top3	MRR
Overview	754	Vote	732	0.7609	0.9290	0.8482	686	0.8163	0.9577	0.8865
		Conf		0.7582	0.9003	0.8407		0.8192	0.9563	0.8908
		CER		0.7117	0.8675	0.8018		0.7624	0.9461	0.8560
		Freq		0.5915	0.7459	0.6950		0.6531	0.8790	0.7728
		Prob		**0.7992**	**0.9331**	**0.8716**		**0.8455**	**0.9636**	**0.9066**
Continent	141	Vote	141	0.9574	1	0.9787	133	0.9699	1	0.9850
		Conf		0.9574	1	0.9787		0.9699	1	0.9850
		CER		0.9362	0.9929	0.9656		0.9549	0.9925	0.9752
		Freq		0.9574	1	0.9787		0.9624	1	0.9812
		Prob		**0.9716**	**1**	**0.9858**		**0.9774**	**1**	**0.9887**
Region	141	Vote	134	0.5448	0.9179	0.7295	115	0.6783	0.9652	0.8232
		Conf		0.4851	0.8060	0.6669		0.6609	0.9478	0.8025
		CER		0.8134	0.9478	0.8835		0.7826	0.9826	0.8812
		Freq		0.4328	0.8134	0.6314		0.6261	0.9478	0.7851
		Prob		**0.6045**	**0.9030**	**0.7599**		**0.7217**	**0.9739**	**0.8442**
SurfaceArea	141	Vote	138	0.7971	0.9565	0.8776	138	0.8261	0.9638	0.8948
		Conf		0.8696	0.9348	0.9102		0.8551	0.9420	0.9076
		CER		0.3768	0.6812	0.5556		0.5217	0.8841	0.7007
		Freq		0.0290	0.1232	0.1856		0.0580	0.5725	0.3475
		Prob		**0.8913**	**0.9710**	**0.9327**		**0.8841**	**0.9710**	**0.9296**
IndepYear	190	Vote	180	0.7111	0.8389	0.7894	171	0.7544	0.9006	0.8240
		Conf		0.6889	0.8111	0.7761		0.7544	0.9240	0.8441
		CER		0.6056	0.7778	0.7148		0.7076	0.9064	0.8158
		Freq		0.6944	0.8167	0.7719		0.7544	0.9123	0.8368
		Prob		**0.7389**	**0.8555**	**0.8146**		**0.7895**	**0.9064**	**0.8605**
Capital	141	Vote	139	0.7986	0.9568	0.8770	129	0.8527	0.9767	0.9154
		Conf		0.7986	0.9712	0.8828		0.8527	0.9767	0.9163
		CER		0.8561	0.9640	0.9142		0.8760	0.9845	0.9302
		Freq		0.7986	0.9496	0.8746		0.8605	0.9767	0.9173
		Prob		**0.7986**	**0.9568**	**0.8764**		**0.8527**	**0.9845**	**0.9143**

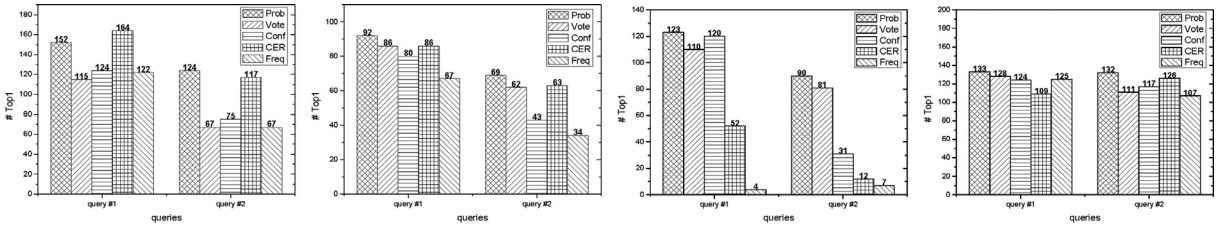


(a) Author (BX) (b) Title (BX) (c) Region (world) (d) SurfaceArea (world)

Fig. 5. Varying the number of analyzed snippets.

Table 8
Different patterns of submitted queries.

Target attribute	Query pattern #1	Query pattern #2
Title (*BX*)	t[*ISBN*]	t[*ISBN*] + t[*Publication Year*] + "title"
Publisher (*BX*)	t[*ISBN*] + t[*Publication Year*] + "publisher"	t[*ISBN*] + t[*Title*] + t[*Publication Year*] + "publisher"
SurfaceArea (*world*)	t[*Name*] + "surface area square km"	t[*Name*] + t[*Continent*] italic>] + t[*Region*] + t[*Capital*] + t[*IndepYear*] + "surface area square km"
IndepYear (*world*)	t[*Name*] + "independence"	t[*Name*] + t[*Continent*] + t[*Region*] + t[*Capital*] + "independence"



(a) *Title* (*BX*, # Testing Tuples = 200) (b) *Publisher* (*BX*, # Testing Tuples = 199) (c) *SurfaceArea* (*world*, # Testing Tuples = 141) (d) *IndepYear* (*world*, # Testing Tuples = 190)

Fig. 6. Varying query patterns (Maximum # of Snippets = 300).

gradually flattens out. The only exception is the *Freq* method applied on the *surface area* attribute of *world*. In this case, due to the presence of trivial numbers, considering more snippets may introduce more noise and compound the difficulty of ranking by frequency. As a result, after 20, the imputation success rate instead decreases with the increasing number of analyzed snippets. It can be observed that *Prob* consistently performs best among the tested approaches.

5.5. Varying query patterns

We also evaluate the performance of the ranking methods given different keyword query patterns. The query patterns for the attributes of *BX* and *world* are presented in Table 8. Generally, the query pattern #1 is short while the query pattern #2 is longer. It can be expected that a short query usually retrieves more snippets than a longer one. The detailed experimental results are presented in Fig. 6. We have the following observations: (1) different query patterns may retrieve different snippets; therefore, they may significantly affect the imputation success rate. Take Fig. 6(a) as an example, the **# Top1** values of *Prob* given a short query and a longer one are 152 and 124 respectively; (2) with either short queries or long queries, *Prob* achieves the best performance among the tested methods; (3) the performance advantage of *Prob* over other approaches are more considerable in the case of longer queries than in the case of short queries. The ranking effectiveness of the *Freq* and *Vote* methods largely depends on the large corpus size of analyzed snippets. Longer queries often result in less snippets being retrieved. In comparison, by semantic reasoning, the *Prob* approach can rank effectively using less snippets. Therefore, the performance advantage of *Prob* tends to increase as the queries become longer.

5.6. With the stricter extraction technique

This subsection evaluates the performance of different ranking approaches with the candidates being extracted by a stricter pattern-based technique. The detailed experimental results on the *BX* and *world* datasets are presented in Tables 6 and 7 respectively. A maximal of 300 snippets are analyzed for each input query. It can be observed that similar to the case of using the loose NER-based extraction technique, the *Prob* approach achieves the best performance among

the tested approaches. As expected, all the tested approaches achieve considerably better ranking accuracy. However, the stricter extraction technique effectively reduces the imputation recall because less values are included in the candidate set.

6. Conclusion

In this paper, we present a semantic and probabilistic ranking framework for web-based relational data imputation. The proposed framework consists of two components, the *snippet influence model* measuring the influence of a retrieved snippet and the *semantic matching model* measuring the semantic similarity between a candidate value in a snippet and a missing attribute value in a relational tuple. We also provide effective estimation solutions for both models. Finally, we demonstrate by extensive experiments on real datasets that the proposed framework performs considerably better than the existing alternatives on ranking accuracy. On future work, it is interesting to investigate whether the proposed probabilistic ranking framework can be extended for the task of web-based question answering. Since the semantic context of a natural language question can be obscure and ambiguous, semantic reasoning is more challenging for question answering than for relational data imputation.

Acknowledgments

This work is supported by the National Basic Research Program of China under grant no. 2012CB316203, the National Natural Science Foundation of China under grant no. 61502390, no. 61332006 and no. 61472321, the NWPU Basic Research Foundation (3102014JSJ0013, 3102014JSJ0005).

References

[1] G.E. Batista, M.C. Monard, A study of k-nearest neighbour as an imputation method, HIS 87 (48) (2002) 251-260.
[2] G. Beydoun, F. García-Sánchez, C.M. Vincent-Torres, A.A. Lopez-Lorca, R. Martínez-Béjar, Providing metrics and automatic enhancement for hierarchical taxonomies, Inf. Process. Manag. 49 (1) (2013) 67–82.
[3] G. Beydoun, G. Low, F. García-Sánchez, R. Valencia-García, R. Martínez-Béjar, Identification of ontologies to support information systems development, Inf. Syst. 46 (2014) 45–60.
[4] S. Buchholz, Using grammatical relations, answer frequencies and the world wide web for TREC question answering, in: TREC, 2001.
[5] Z. Chen, J. Li, C. Jiang, H. Liu, Q. Chen, Z. Li, A context-aware entity ranking method for web-based data imputation, Chin. J. Comput. 38 (9) (2015) 1755–1766.
[6] C.L. Clarke, G.V. Cormack, T.R. Lynam, Exploiting redundancy in question answering, in: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2001, pp. 358–365.
[7] A.R.T. Donders, G.J. van der Heijden, T. Stijnen, K.G. Moons, Review: a gentle introduction to imputation of missing values, J. Clin. Epidemiol. 59 (10) (2006) 1087–1091.
[8] S. Dumais, M. Banko, E. Brill, J. Lin, A. Ng, Web question answering: Is more always better? in: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2002, pp. 291–298.
[9] W. Fan, F. Geerts, Foundations of data quality management, Synth. Lect. Data Manag. 4 (5) (2012) 93–118.
[10] J.W. Grzymala-Busse, M. Hu, A comparison of several approaches to missing attribute values in data mining, in: Rough Sets and Current Trends in Computing, Springer, 2001, pp. 378–385.
[11] J. Han, M. Kamber, J. Pei, Data Mining:Concepts and Techniques, Elsevier, 2011.
[12] J.J. Hox, A review of current software for handling missing data, Kwantitatieve Methoden 62 (1999) 123–138.
[13] C. Kwok, O. Etzioni, D.S. Weld, Scaling question answering to the web, ACM Trans. Inf. Syst. (TOIS) 19 (3) (2001) 242–262.
[14] X. Li, W. Meng, C. Yu, T-verifier: Verifying truthfulness of fact statements, in: Proceedings of the IEEE 27th International Conference on Data Engineering (ICDE), IEEE, 2011, pp. 63–74.
[15] Z. Li, M. Sharaf, L. Sitbon, X. Du, X. Zhou, et al., Core: a context-aware relation extraction method for relation completion, IEEE Trans. Knowl. Data Eng. 26 (4) (2014b) 836–849.
[16] Z. Li, M.A. Sharaf, L. Sitbon, S. Sadiq, M. Indulska, X. Zhou, A web-based approach to data imputation, World Wide Web 17 (5) (2014a) 873–897.
[17] J. Lin, B. Katz, Question answering from the web using knowledge annotation and knowledge mining techniques, in: Proceedings of the Twelfth International Conference on Information and Knowledge Management, ACM, 2003, pp. 116–123.
[18] A. Mikheev, M. Moens, C. Grover, Named entity recognition without gazetteers, in: Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, 1999, pp. 1–8.
[19] Z. Nie, J.-R. Wen, W.-Y. Ma, Object-level vertical search, in: CIDR, 2007, pp. 235–246.
[20] Z. Nie, Y. Zhang, J.-R. Wen, W.-Y. Ma, Object-level ranking: bringing order to web objects, in: Proceedings of the 14th International Conference on World Wide Web, ACM, 2005, pp. 567–574.
[21] L. Page, S. Brin, R. Motwani, T. Winograd, in: The Pagerank Citation Ranking: Bringing Order to the Web, Stanford InfoLab, 1999.
[22] D. Ravichandran, E. Hovy, Learning surface text patterns for a question answering system, in: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2002, pp. 41–47.
[23] E. Riloff, Automatically generating extraction patterns from untagged text, in: Proceedings of the National Conference on Artificial Intelligence, 1996, pp. 1044–1049.
[24] N. Schlaefer, P. Gieselmann, T. Schaaf, A. Waibel, A pattern learning approach to question answering within the Ephyra framework, in: Text, Speech and Dialogue, Springer, 2006, pp. 687–694.
[25] N. Schlaefer, J. Ko, J. Betteridge, M.A. Pathak, E. Nyberg, G. Sautter, Semantic extensions of the Ephyra QA system for TREC 2007, in: TREC, 2007.
[26] G. Shu, O.F. Rana, N.J. Avis, C. Dingfang, Ontology-based semantic matchmaking approach, Adv. Eng. Softw. 38 (1) (2007) 59–67.
[27] D. Shukla, R. Singhai, N.S. Thakur, N. Dembla, Some imputation methods to treat missing values in knowledge discovery in data warehouse, Int. J. Data Eng. (IJDE) 1 (2) (2010).
[28] S. Song, A. Zhang, L. Chen, J. Wang, Enriching data imputation with extensive similarity neighbors, Proc. VLDB Endow. 8 (11) (2015) 1286–1297.
[29] H. Sun, N. Duan, Y. Duan, M. Zhou, Answer extraction from passage graph for question answering, in: Proceedings of the Twenty-Third International joint Conference on Artificial Intelligence, AAAI Press, 2013, pp. 2169–2175.
[30] R.C. Wang, W.W. Cohen, Language-independent set expansion of named entities using the web, in: Proceedings of the Seventh IEEE International Conference on Data Mining, ICDM, IEEE, 2007, pp. 342–350.
[31] W. Wang, C. Xiao, X. Lin, C. Zhang, Efficient approximate entity extraction with edit distance constraints, in: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, ACM, 2009, pp. 759–770.

[32] S. Wu, X. Feng, Y. Han, Q. Wang, Missing categorical data imputation approach based on similarity, in: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2012, pp. 2827–2832.
[33] J. Yu, D. Tao, M. Wang, Y. Rui, Learning to rank using user clicks and visual features for image retrieval, Cybern. IEEE Trans. 45 (4) (2015) 767–779.
[34] J. Yu, D. Tao, J. Li, J. Cheng, Semantic preserving distance metric learning and applications, Inf. Sci. 281 (2014) 674–686.
[35] S. Zhang, J. Zhang, X. Zhu, Y. Qin, C. Zhang, Missing value imputation based on data clustering, in: Transactions on Computational Science I, Springer, 2008, pp. 128–138.
[36] C.N. Ziegler, S.M. Mcnee, J.A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: WWW, 2005, pp. 22–32.