Data Cleaning via Question Asking

Xinyang Zhang
Lehigh University
xizc15@lehigh.edu

Yujie Ji
Lehigh University
yuj216@lehigh.edu

Chanh Nguyen
Lehigh University
cpn217@lehigh.edu

Ting Wang
Lehigh University
inbox.ting@gmail.com

ABSTRACT

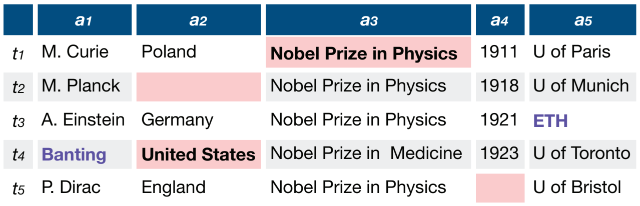
As one critical task in the data analysis pipeline, data cleaning is notoriously human labor-intensive and error-prone. Knowledge base-assisted data cleaning has proved a powerful tool for finding and fixing data defects; however, its applicability is inevitably bounded by the natural limitations of knowledge bases. Meanwhile, although a vast number of knowledge sources exist in the form of free-text corpora (e.g., Wikipedia), transforming them into formats usable by existing data cleaning tools can be prohibitively costly and error-prone, if not at all impossible.

Here, we present DeepClean, the first end-to-end data cleaning framework powered by free-text knowledge sources. At a high level, DeepClean leverages a knowledge source through its question-answering (QA) interface and achieves high-quality cleaning via iterative question asking. Specifically, DeepClean detects and repairs data defects in three stages: (i) Pattern extraction - it automatically discovers the semantic types of the data attributes as well as their correlations; (ii) Question generation - it translates each data tuple into a minimal set of validation questions; (iii) Completion and repair - by checking the answers returned by the knowledge source against the data values, it identifies erroneous cases and suggests possible fixes. Through extensive empirical studies, we demonstrate that DeepClean is applicable to a range of domains, and can effectively repair a variety of data defects, highlighting data cleaning powered by free-text knowledge sources as a promising direction for future research.

ACM Reference Format:
Xinyang Zhang, Yujie Ji, Chanh Nguyen, and Ting Wang. 2018. Data Cleaning via Question Asking. In *Proceedings of KDD 2018 Workshop on Interactive Data Exploration and Analytics (IDEA'18) (IDEA @ KDD'18).* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

Real-world datasets often contain various types of defects (e.g., inaccurate, missing, or duplicate values). Data cleaning, the process of detecting and fixing data defects, is one critical yet still overlooked task in the data analysis pipeline. Indeed, it was estimated that on average data scientists spend over 50 percent of their time on massaging and cleaning unruly data, before it can be explored for useful insights [17].



	a_1	a_2	a_3	a_4	a_5
t_1	M. Curie	Poland	**Nobel Prize in Physics**	1911	U of Paris
t_2	M. Planck		Nobel Prize in Physics	1918	U of Munich
t_3	A. Einstein	Germany	Nobel Prize in Physics	1921	**ETH**
t_4	**Banting**	**United States**	Nobel Prize in Medicine	1923	U of Toronto
t_5	P. Dirac	England	Nobel Prize in Physics		U of Bristol

Figure 1: A relational table $\mathcal{T}$ of Nobel Laureates, with missing and erroneous values highlighted.

Recently, knowledge base-assisted data cleaning (e.g., [8]) has emerged as a promising approach for such tasks. Intuitively, it performs cleaning by aligning the (dirty) data with publicly available knowledge bases (e.g., Freebase [5], DBPedia [19], and Yago [14]). Compared with prior art (e.g., [4, 12, 24, 25, 29]), it significantly improves the repair accuracy and reduces the requirement for external resources (e.g., domain expertise [34] and master data [11]). Yet, the applicability of this approach is inevitably bounded by the natural limitations of knowledge bases, such as: (i) their fixed schema are not expressive to describe many complicated relationships, (ii) they are sparsely populated with respect to many domains, and (iii) they often contain stale information, due to infrequent update cycles.

Meanwhile, a vast number of knowledge sources exist in the form of free-text encyclopedias (e.g., Wikipedia), of which content quality and format is strictly governed by detailed guidelines. Moreover, such knowledge sources are constantly maintained and updated. Compared with traditional knowledge bases, free-text encyclopedias often provide larger coverage, richer textual representation, and more accurate information [32]. However, while knowledge bases are designed for machines to process, free-text encyclopedias are designed for humans to read. Transforming them into formats directly usable for existing data cleaning tools can be prohibitively costly and error-prone, if not at all impossible.

Interestingly, to date a variety of question-answering (QA) modules (e.g., [6, 10, 31]) have been built for free-text encyclopedias to automatically answer natural language questions (e.g., *"What Nobel Prize was awarded to Marie Curie in 1911?"*). Built using deep neural networks (e.g., LSTM) and trained using large question-answer corpora (e.g., the SQuAD dataset [22]), these modules provide QA capabilities comparable to human level comprehension[1]. Thus, in this paper, we pose the following question:

IDEA @ KDD'18, August 20th, 2018, London, United Kingdom

[1] As of January 22 2018, the R-NET+ [30] model has outperformed humans on the SQuAD task in terms of ExactMatch score (http://stanford-qa.com/).

> *Can we achieve high-quality data cleaning by leveraging a free-text encyclopedia via its QA interface?*

Challenges. While conceptually simple, achieving this goal represents a set of non-trivial challenges.

First, translating the data cleaning task into a sequence of question asking and answer checking is challenging. It requires comprehensive understanding of the data semantics (e.g., attribute types and correlations). Yet, in realistic settings, we often lack reliable, meaningful labeling of the data.

Second, multiple types of defects (e.g., missing and erroneous values) are often intertwined in the data. As the data attributes are interdependent, repairing one value often depends on the presence and correctness of other values. This leads to the challenging problem of scheduling the validation order for different values.

Last but not least, to scale up to large number of attributes and tuples, it is desirable to minimize the number of questions issued to the QA interface.

Our Work. Here, we present the design, implementation, and evaluation of DeepClean, the first end-to-end data cleaning framework powered by free-text knowledge sources. At a high level, DeepClean achieves high-quality data cleaning in three major stages, as illustrated in Figure 2.

(i) *Pattern Extraction* - We assume a "cold-start" setting wherein the data semantics is unknown a priori. In this stage, DeepClean automatically discovers the attribute types and correlations by bridging the data with the knowledge source through its QA interface, without requiring any manual annotations.

(ii) *Question Generation* - Based on the discovered data semantics, DeepClean defines a set of question templates, each designed to verify one attribute. Using these templates, DeepClean is able to generate a set of validation questions for each data tuple.

Example 1.1. In Figure 1, to verify the third attribute of the first tuple in the table (i.e., *"Nobel Prize in Physics"*), DeepClean generates the question as: *"Which Nobel Prize for Laureate M. Curie and Year 1911?"*.

(iii) *Completion and Repair* - Finally, DeepClean performs cleaning by iteratively issuing validation questions to the QA interface, checking returned answers against data values, identifying defects, and suggesting possible fixes. To maximize the cleaning effectiveness and to minimize the QA overhead, DeepClean carefully schedules the execution order of validation questions.

Example 1.2. To the question in Example 1.1, the QA interface returns the answer of *"she won the 1911 Nobel Prize in Chemistry"*, with the most likely answer underlined. Given the conflict between this answer and $t_1[a_3]$, an error is pinpointed and the possible repair of *"Nobel Prize in Chemistry"* is suggested.

We prototype DeepClean and evaluate its efficacy using a variety of real datasets. It is shown that DeepClean is applicable to a range of domains, and can effectively identify and fix a variety of data defects (including missing and erroneous values), thanks to the extensive coverage and rich contextual evidence bestowed by free-text knowledge sources.

Contributions. The main contributions of this paper can be summarized as follows.

- We envision the paradigm of data cleaning by exploiting vastly available free-text knowledge sources through QA interfaces, and present DeepClean, the first end-to-end design that realizes this paradigm.
- We prototype DeepClean and propose a suite of optimization strategies that significantly improve its usability and applicability in realistic settings.
- We conduct extensive empirical evaluation on the efficacy of DeepClean using a variety of real datasets. The results highlight data cleaning powered by free-text knowledge sources as a promising direction for further research.

It is worth emphasizing that we are not arguing to replace existing data cleaning tools with DeepClean. Indeed, as shown in our empirical evaluation, DeepClean well complements existing tools, especially in repairing values that exist in complicated interdependencies with other values. Thus, integrating DeepClean with these tools is a future research topic with strong practical relevance.

2 OVERVIEW OF DEEPCLEAN

In this section, we motivate the overall design of DeepClean.

2.1 Preliminaries

We first introduce a set of fundamental concepts and assumptions used throughout the paper.

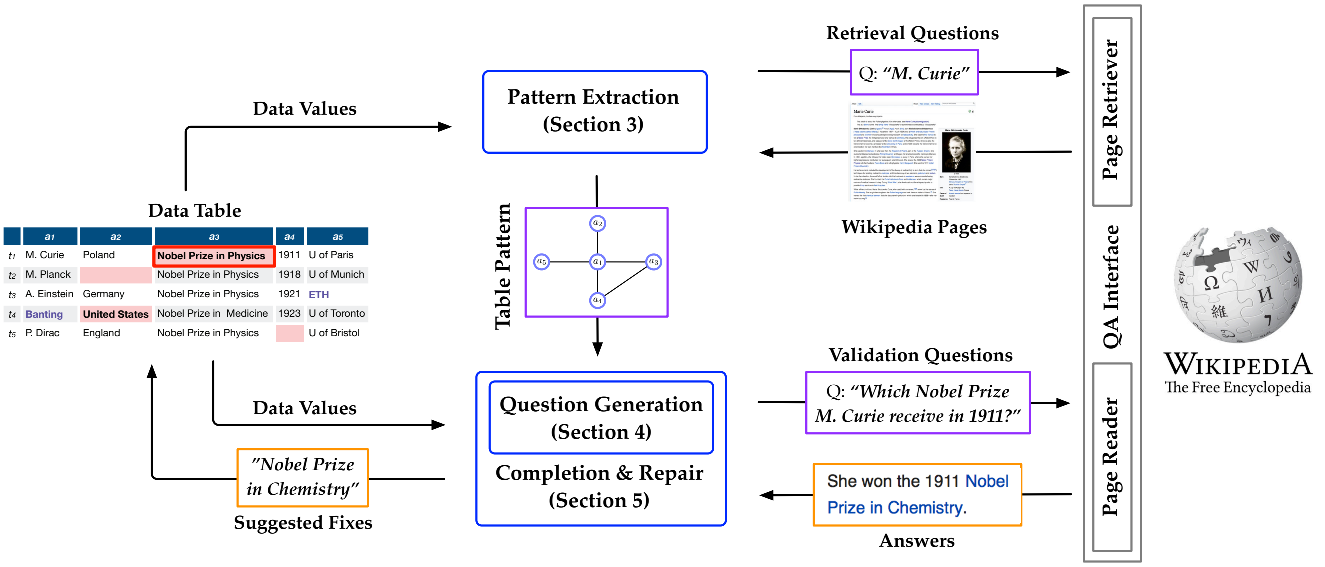
Data Table We assume that the (dirty) data is stored in a tabular form, which is one of the most widely used data formats (e.g., SQL databases, Web tables, spreadsheets). In general, a table $\mathcal{T}$ describes one type of entities (e.g., *"Nobel Laureate"*); each row (tuple) of $\mathcal{T}$ represents one instance of that entity type (e.g., *"Albert Einstein"*), and each column of $\mathcal{T}$ represents one attribute of that entity type (e.g., *"Alma mater"*). A schematic table is shown in Figure 1. For simplicity, we assume a single table $\mathcal{T}$ in the data, while the discussion can be generalized to the case of multiple tables.

Let $\mathcal{A}$ be the set of attributes in $\mathcal{T}$. Given a tuple $t \in T$, its value with respect to an attribute $a \in \mathcal{A}$ is called a *cell*, denoted by $t[a]$.

Table Pattern. We assume a cold-start scenario in which $\mathcal{T}$'s semantics is unknown a priori. There are two types of semantics that DeepClean is designed to uncover: (i) the semantic type $\text{tp}(a)$ of each attribute $a \in \mathcal{A}$, which is defined as its categorization in the free-text knowledge source, and (ii) a's correlation with other attributes $\mathcal{A} \setminus \{a\}$, which is defined as the subset of $\mathcal{A} \setminus \{a\}$, $\text{cr}(a)$, that are highly correlated with a.

Example 2.1. In Figure 1, $\text{tp}(a_1)$ = *"Laureate"*, $\text{tp}(a_2)$ = *"Nobel Prize"*, and $\text{tp}(a_3)$ = *"Year"*; $\text{cr}(a_3) = \{a_1, a_4\}$, because the type of Nobel Prize (a_3) is highly correlated with both the laureate (a_1) and the award year (a_3).

We refer to this collection of type and correlation information $\mathcal{G} = \{\text{tp}(a), \text{cr}(a)\}_{a \in \mathcal{A}}$ as $\mathcal{T}$'s *table pattern*. For example, the pattern of the table in Figure 1 is illustrated in Figure 3.



	a_1	a_2	a_3	a_4	a_5
t_1	M. Curie	Poland	**Nobel Prize in Physics**	1911	U of Paris
t_2	M. Planck		Nobel Prize in Physics	1918	U of Munich
t_3	A. Einstein	Germany	Nobel Prize in Physics	1921	**ETH**
t_4	**Banting**	**United States**	Nobel Prize in Medicine	1923	U of Toronto
t_5	P. Dirac	England	Nobel Prize in Physics		U of Bristol

Figure 2: Illustration of DeepClean framework.

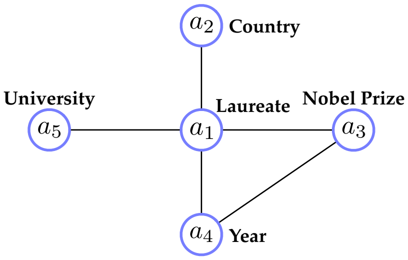


Figure 3: The table pattern of $\mathcal{T}$ in Figure 1.

Free-Text Knowledge Source. In this paper, we use Wikipedia as a concrete instance of free-text knowledge sources, due to its popularity and public nature.

Wikipedia is an online encyclopedia, of which content quality and format is strictly governed by detailed guidelines. Unlike traditional knowledge bases, Wikipedia articles are constantly updated, making it an ideal knowledge source for data cleaning tasks.

Wikipedia is organized as a repository of *pages*: each page describes one distinct *topic* (e.g., a person, a place, an organization), while pages may be linked through page-links [1]. In addition to topic pages, Wikipedia also maintains "redirect" pages, which list alternative representations of given topics (e.g., *"Marie Curie"* versus *"Marie Sklodowska-Curie"*), and "disambiguation" pages, which list pages of topics possibly referred to by ambiguous terms (e.g., *Marie Curie* may refer to either a physicist or a Polish film).

To organize the vast number of pages, Wikipedia also maintains a hierarchical category system, in which each category groups together pages on similar topics. Note that each page is possibly associated with multiple categories. The fields of a Wikipedia page used in DeepClean are summarized in Table 1.

Question Answering. To make better use of free-text knowledge sources, a plethora of question-answering (QA) modules have been built to automatically answer natural language questions posed by users (e.g., "*Which year was Albert Einstein awarded the Nobel Prize?*"). For Wikipedia, a variety of QA modules have been proposed (e.g., [6, 10, 31]). Built upon deep neural networks (e.g., bidirectional LSTM) and trained using large question-answer corpora (e.g., the SQuAD dataset [22]), these modules provide QA capabilities comparable to human level comprehension.

Field	Definition
title	unique name
text	detailed description
category	categorization
page-link	hyperlinks to other pages

Table 1. Fields of a Wikipedia page used in DeepClean.

Despite their significant variations, all the QA modules provide two fundamental functions: (i) page retriever - given a retrieval query q (e.g., *"M. Curie"*), it extracts a set of Wikipedia pages most relevant to q; (ii) page reader - given a natural language question q (e.g., "*Which year was M. Curie awarded the Nobel Prize?*", it predicts the *span* (i.e., a small piece of text within a Wikipedia page) in which the answer to q most likely lies. For example, to the question above, the QA module may predict the span of "*she won the 1911 Nobel Prize in Chemistry*".

In implementing DeepClean, we treat the QA module as a black box. Thus, the concrete QA module is immaterial, provided that it offers accurate prediction for answer spans. In the following, we instantiate the QA module with the DrQA model, which achieves F1 score of 79.353 on the SQuAD dataset [6].

2.2 DeepClean in A Nutshell

Intuitively, DeepClean performs high-quality data cleaning by translates this task into a sequence of question asking and answer checking via the QA interface of the knowledge source. As illustrated in Figure 2, the high-level design of DeepClean consists of three major stages, namely, pattern extraction, question generation, and completion and repair.

Pattern Extraction. To make DeepClean applicable to realistic settings, we assume that $\mathcal{T}$'s semantics (attribute types and correlations) is completely unknown. In this stage, DeepClean automatically discovers $\mathcal{T}$'s table pattern $\mathcal{G}$ by bridging $\mathcal{T}$ with the knowledge source through its QA interface. Specifically, to identify attribute types, we propose an instance-based "bootstrapping" approach; to find attribute correlations, we apply a weakly supervised reward-guided search.

Question Generation. According to $\mathcal{T}$'s table pattern $\mathcal{G}$, DeepClean defines a set of question templates, each designed to verify one attribute. Note that one attribute may be associated with multiple question templates, which enables (i) higher chance to find relevant information in the knowledge source and (ii) more flexibility in scheduling the question execution order in the next stage. Based on such question templates, for each tuple, DeepClean generates a set of validation questions to verify each of its cells.

Completion and Repair. In this stage, DeepClean compares answers returned by the knowledge source against actual cell values. If conflicts are found, DeepClean pinpoints errors and further suggest possible repairs (e.g., the most likely answer found in the knowledge source).

To maximize the repair effectiveness and to minimize the number of questions issued to the knowledge source, DeepClean carefully schedules the question execution order by prioritizing questions whose answers maximally reduce a tuple's overall uncertainty.

Note that in addition to suggesting possible repairs, DeepClean also provides the contexts within which the repairs are found (i.e., the answer spans). Such contextual information can be valuable for the data analysts to make final repair decisions.

In § 3, § 4, and § 5, we elaborate on each of DeepClean's core components.

3 PATTERN EXTRACTION

We assume a "cold-start" scenario in which $\mathcal{T}$'s semantics is either unavailable or unusable (an example is shown in Figure 1), which is especially true for data constructed from Web resources where cryptic naming conventions are often used. In the stage of pattern extraction, DeepClean automatically discovers $\mathcal{T}$'s semantics by bridging it with knowledge source through its QA interface.

Following existing work (e.g., [8, 12, 25]), our running assumption is as follow: in realistic settings, the number of erroneous tuples is relatively small, compared with the total number of tuples in $\mathcal{T}$; thus, the aggregated information across a majority of the tuples tend to be accurate.

We divide pattern extraction into two subtasks, discovering attribute types and discovering attribute correlations, detailed in § 3.1 and § 3.2 respectively.

3.1 Discovering Attribute Types

Let $\mathcal{A}$ denote $\mathcal{T}$'s attributes. We define $\mathcal{A}$'s semantic types using Wikipedia categories (e.g., *"Nobel Prize"*). To discover $\mathcal{A}$'s semantic types, we adopt an instance-based "bootstrapping" approach, which does not require any manual annotations. Intuitively, for each attribute $a \in \mathcal{A}$, we find the set of Wikipedia pages $\mathcal{P}(a)$ most relevant to a and infer a's Wikipedia categories by aggregating the categories of the pages in $\mathcal{P}(a)$.

Disambiguation. For each cell value $t[a]$ of tuple $t \in \mathcal{T}$, we fetch the set of Wikipedia pages $\mathcal{P}(t[a])$ related to $t[a]$ through the page retriever of the QA interface (§ 2.1). We then classify $t[a]$ into three possible classes.

- Topic value, which corresponds to a specific topics in Wikipedia (e.g., *"Nobel Prize in Physics"*). In this case, $\mathcal{P}(t[a])$ contains one unique page as $t[a]$'s topic page, denoted by p_a^t.
- Ambiguous value, which potentially refers to multiple topics in Wikipedia (e.g., *"Banting"*). In this case, $\mathcal{P}(t[a])$ contains more than one page. To find the exact one pointed to by $t[a]$, a disambiguation process is necessary.
- Literal value, which is neither a topic nor an ambiguous value. In this case, $\mathcal{P}(t[a])$ is empty. Literal values tend to appear in the text of other topic pages.

For an ambiguous value $t[a]$, we perform two disambiguation operations to identify its exact topic page among the set of candidates $\mathcal{P}(t[a])$. Recall that each Wikipedia page p is associated with a set of fields (see Table 1). Below we use $C(p)$ and $\mathcal{L}(p)$ to denote p's categories and page-links respectively.

Algorithm 1: Discovering Attribute Types

```
Input: relevant Wikipedia pages $\{\mathcal{P}(t[a])\}_{t \in \mathcal{T}, a \in \mathcal{A}}$
Output: $\mathcal{A}$'s semantic types
// disambiguation
$\mathcal{L}_{\text{queued}} \leftarrow \{t[a]\}_{t \in \mathcal{T}, a \in A}$;
while not converged do
    for $t[a] \in \mathcal{L}_{queued}$ do
        for each $p \in \mathcal{P}(t[a])$ do
            // attribute- and tuple-filtering
            compute a-score($p$) and t-score($p$);
            if a-score($p$) = 0 or t-score($p$) = 0 then
                remove $p$ from $\mathcal{P}(t[a])$;
        if $p_a^t$ is found then remove $t[a]$ from $\mathcal{L}_{\text{queued}}$;
// categorization
for each $a \in \mathcal{A}$ do
    $\mathcal{T}_a \leftarrow \{t \in \mathcal{T} | p_a^t \text{ is identified}\}$;
    if $\frac{|\mathcal{T}_a|}{|\mathcal{T}|} > 0.5$ then
        for $t \in \mathcal{T}_a$ do populate $C(p_a^t)$ ;
        pick $c^* = \arg\max_c$ tf-idf($c, a$);
    else pick $c^*$ = the majority NER type of $\{t[a]\}_{t \in \mathcal{T}}$ ;
```

[*Attribute-Level Filtering*] In this operation, we examine the cell values $\{t'[a]\}_{t'}$ of other tuples $t' \in \mathcal{T}$. Let $\mathcal{T}'$ represent the subset of tuples such that for each $t' \in \mathcal{T}'$, $t'[a]$ is a topic value. We then identify the set of categories shared by their corresponding topic pages, denoted by $\bigcap_{t' \in \mathcal{T}'} C(p_a^{t'})$. Among $\mathcal{P}(t[a])$, we search for the pages whose categories maximally match $\bigcap_{t' \in \mathcal{T}'} C(p_a^{t'})$, as measured by their attribute-level scores:

$$\text{a-score}(p) = \frac{\left|\left(\bigcap_{t' \in \mathcal{T}'} C(p_a^{t'})\right) \cap C(p)\right|}{\left|\bigcap_{t' \in \mathcal{T}'} C(p_a^{t'})\right|} \tag{1}$$

where p is a page in $\mathcal{P}(t[a])$.

Example 3.1. In Figure 1, as a surname, $t_4[a_1]$ (*"Banting"*) is an ambiguous value, possibly referring to multiple notable figures. Yet, all of $t_1[a_1]$, $t_2[a_1]$, $t_3[a_1]$, and $t_5[a_1]$ are topic values, which share the category of *"Nobel Laureates"*. Among $t_4[a_1]$'s candidate pages, only the page of *"Frederick Banting"* matches this category, which we consider as $t_4[a_1]$'s topic page.

[*Tuple-Level Filtering*] In this operation, to disambiguate $t[a]$, we examine t's values with respect to attributes other than a. Let $\mathcal{A}'$ be the subset of attributes such that $t[a']$ is a topic value for $a' \in \mathcal{A}'$ and $\mathcal{L}(p^t_{a'})$ be the set of pages pointed by some page-links in $p^t_{a'}$. Then among $\mathcal{P}(t[a])$, we search for the pages that frequently appear in the sets $\{\mathcal{L}(p^t_{a'})\}_{a' \in \mathcal{A}'}$, as measured by their tuple-level scores:

$$\text{t-score}(p) = \frac{\sum_{a' \in \mathcal{A}'} \mathbb{I}_{\mathcal{L}(p^t_{a'})}(p)}{|\mathcal{A}'|} \tag{2}$$

where $\mathbb{I}_\Omega(x)$ is an indicator function, which returns 1 if $x \in \Omega$ and 0 otherwise.

Example 3.2. In Figure 1, $t_3[a_5]$ is an ambiguous value (e.g., it may potentially refer to Ethiopia), while $t_3[a_1]$ is a topic value. Among $t_3[a_5]$'s candidate pages, only the page of *"ETH Zurich"* appears in $\mathcal{L}(p^{t_3}_{a_5})$, which is thus considered as $t_3[a_5]$'s topic page.

In practice, we interleave attribute-level filtering with tuple-level filtering to maximize the effectiveness of disambiguation. Intuitively, as more topic values are uncovered, there emerge more opportunities to apply the attribute- or tuple-level filtering.

Categorization. After maximally reducing the number of ambiguous values, we proceed to discovering the attribute types. Given attribute a, let $\mathcal{T}_a$ be the subset of tuples such that for $t \in \mathcal{T}_a$, its topic page p^t_a is identified. We find a's semantic type as the optimal category by aggregating the categories of all the pages $\{p^t_a\}_{t \in \mathcal{T}_a}$.

Recall that the categories in Wikipedia form a hierarchy, while each page is likely associated with multiple categories. To select the optimal category, we preform the following procedure.

(1) For each $t \in \mathcal{T}_a$, we first populate its category set $C(p^t_a)$ by including all their ancestor categories in the hierarchy (e.g., *"Nobel Laureates"* as an ancestor category of *"Nobel Laureates in Physics"*);

(2) We then compute the term frequency-inverse document frequency score $\text{tf-idf}(c, a) = \text{tf}(c, a) \cdot \text{idf}(c)$ for each category c. The term frequency $\text{tf}(c, a)$ measures how frequently c appears in $\{C(p^t_a)\}_{t \in \mathcal{T}_a}$, while the inverse document frequency $\text{idf}(c)$ measures how important c is. Specifically, we define $\text{tf}(c, a)$ and $\text{idf}(c)$ as follows:

$$\text{tf}(c, a) = \frac{\sum_{t \in \mathcal{T}_a} \mathbb{I}_{C(p^t_a)}(c)}{|\mathcal{T}_a|} \qquad \text{idf}(c) = \log \frac{n_{\text{total}}}{n_c}$$

where n_{total} is the total number of pages in Wikipedia and n_c is the number of pages associated with c.

(3) Finally, we pick the category c^* with the highest tf-idf score as a's semantic type.

In the case that most of $\{t[a]\}_{t \in \mathcal{T}}$ are literal values, we resort to named entity recognition (NER) categories [20]. To detect $t[a]$'s NER category, we search for $t[a]$ in the topic pages of other cell values $\{t[a']\}_{a' \neq a}$ and apply the NER extractor of the Stanford OpenIE library [2]. We define a's semantic type as the majority NER category among $\{t[a]\}_{t \in \mathcal{T}}$.

Putting everything together, Algorithm 1 sketches the procedure of discovering $\mathcal{A}$'s semantic types.

3.2 Discovering Attribute Correlations

As will be shown in § 4, to generate questions effective for data cleaning, it is crucial to understand the correlations between different attributes. For given attribute $a^* \in \mathcal{A}$, we are interested in finding a minimal subset of $\mathcal{A} \setminus \{a^*\}$, denoted by $\text{cr}(a^*)$, such that for tuple $t \in \mathcal{T}$, $t[\text{cr}(a^*)]$ (i.e., t's values projected on $\text{cr}(a^*)$) and $t[a^*]$ appear in the same semantic context (e.g., the same sentence) with high probability. We refer to $\text{cr}(a^*)$ as a^*'s correlation set. Further, by aggregating such correlation sets, we group $\mathcal{A}$ into a set of "cliques", similar to undirected graphical models.

Example 3.3. In Figure 1, a_3 represents the type of Nobel Prize, which is highly correlated with a_1 (*"Laureate"*) and a_4 (*"Year"*), while a_1 or a_4 alone is insufficient to determine a_3 (e.g., M. Curie won the Nobel Prize in Physics in 1903 and the Nobel Prize in Chemistry in 2011). Thus, $\text{cr}(a_3) = \{a_1, a_4\}$.

Reward-Guided Search. To find the optimal $\text{cr}(a^*)$ for given attribute a^*, we apply a weakly supervised reward-guided search. We model the search as a sequence of actions of adding one attribute a time from $\mathcal{A} \setminus \{a\}$ to a set $\mathcal{A}_{a^*}$ (which is initialized as an empty set), and define the state space to be all the possible assignments of $\mathcal{A}_{a^*}$ (i.e., all the subsets of $\mathcal{A} \setminus \{a\}$). Each action moves the search from one state to another.

The *value* function measures $\mathcal{A}_{a^*}$'s quality, which is defined in a recursive form:

$$\begin{cases} \mathsf{v}(\mathcal{A}_{a^*} \bigcup \{a\}) = \mathsf{v}(\mathcal{A}_{a^*}) + \pi(\mathcal{A}_{a^*}, a) \\ \mathsf{v}(\emptyset) = 0 \end{cases} \tag{3}$$

where a represents the attribute added to $\mathcal{A}_a$ and the *policy* function $\pi(\mathcal{A}_{a^*}, a)$ scores the action of adding a to $\mathcal{A}_{a^*}$. Note that we disallow adding duplicate attributes to $\mathcal{A}_{a^*}$.

We use a beam search strategy to find a state with the highest value, which represents the optimal assignment of $\text{cr}a^*$.

Implementation. Next we detail the implementation of this search strategy.

Intuitively, we measure $\mathcal{A}_{a^*}$'s quality with respect to a^* as follows: for tuple $t \in \mathcal{T}$, we generate a question asking for $t[a^*]$ conditioned on $t[\mathcal{A}_{a^*}]$ (the details of question generation are deferred to § 4); if the QA interface is able to find the correct answer with high probability, we consider that a^* is highly correlated with $\mathcal{A}_{a^*}$.

Thus we measure the value function $\mathsf{v}(\mathcal{A}_{a^*})$ as the overall accuracy of retrieving correct answers for $t[a^*]$ given $t[\mathcal{A}_{a^*}]$ for each $t \in \mathcal{T}$, while the policy function $\pi(\mathcal{A}_{a^*}, a)$ is trivially the change of accuracy due to adding a to $\mathcal{A}_{a^*}$. Using the beam search strategy, the search for optimal $\mathcal{A}_{a^*}$ is reduced to a sequence of question asking and answer checking.

4 QUESTION GENERATION

Next we show how to generate validation questions to verify each value of $\mathcal{T}$.

Let $\mathcal{G} = \{\text{tp}(a), \text{cr}(a)\}_a$ denote $\mathcal{T}$'s table pattern. For given tuple t and attribute a^*, we verify the value of $t[a^*]$ by generating one or more validation questions (which is dependent on the number of correlation sets found for a^*). For simplicity of discussion, here we assume $\mathrm{cr}(a^*)$ contains only one correlation set.

In our empirical evaluation, we find the following question template particularly effective for extracting relevant answers. The question template comprises a header and a body:

$$\text{question}(a^*, \mathrm{cr}(a^*), t) \rightarrow \text{header}(a^*) \text{ FOR body}(\mathrm{cr}(a^*), t)$$

Specifically, the header header(a^*) is generated from a^* using the following rule:

$$\begin{array}{lcll} \text{header}(a^*) & \rightarrow & \textit{Which } \mathrm{tp}(a^*) & (\mathrm{tp}(a^*) \in \text{Wikipedia}) \\ & \mid & \textit{Where} & (\mathrm{tp}(a^*) = \text{LOCATION}) \\ & \mid & \textit{When} & (\mathrm{tp}(a^*) = \text{TIME}) \\ & \mid & \textit{Who} & (\mathrm{tp}(a^*) = \text{PERSON}) \\ & \mid & \textit{What} & (\mathrm{tp}(a^*) = \text{MISC}) \end{array}$$

Intuitively, if a^*'s semantic type $\mathrm{tp}(a^*)$ is described by Wikipedia categories, we use "*which* $\mathrm{tp}(a^*)$" as the question header; if $\mathrm{tp}(a^*)$ is described by NER categories [20], we use the corresponding question word directly.

The question body body($\mathrm{cr}(a^*)$) is generated from $\mathrm{cr}(a^*)$ and t in a recursive manner:

$$\text{body}(\mathrm{cr}(a^*), t) \rightarrow \mathrm{tp}(a)\, t[a] \text{ AND body}(\mathrm{cr}(a^*) \setminus \{a\})$$

Note that here we assume the attribute order in $\mathrm{cr}(a^*)$ is already optimized as in § 3.2.

Example 4.1. In Figure 1, with $a^* = a_3$, $\mathrm{cr}(a^*) = \{a_1, a_4\}$, and $t = t_1$, following the rules above, DeepClean generates the question of *"Which Nobel Prize* for *Laureate M. Curie* and *Year 1911"*.

5 COMPLETION AND REPAIR

In the stage, DeepClean performs effective data cleaning by iteratively (i) submitting validation questions to the QA interface, (ii) comparing returned answers against data values, and (iii) identifying defects and suggesting possible fixes.

In DeepClean, we consider two types of defects, missing values and erroneous values, which often co-exist in the data. Because fixing a given attribute value depends on the existence and correctness of other attribute values, it is crucial to carefully schedule the execution order of validation questions.

A Unified Framework. Without loss of generality, consider a given tuple $t \in \mathcal{T}$ that contains both missing and erroneous values. Note that while missing values are obvious, erroneous values need to be detected. We present a labeling-based algorithm that fills t's missing values (*completion*) and corrects t's erroneous values (*repair*) in a unified framework, as sketched in Algorithm 2.

Specifically, when attribute a^*'s correlation set $\mathrm{cr}(a^*)$ is empty, there is no rule available to fix $t[a^*]$, we thus assume $t[a^*]$ is either correct by default if $t[a^*]$ is present, or unfixable if $t[a^*]$ is missing (line 3-6).

We fix the remaining attribute values in an iterative manner. if for attribute a^*, the values of $\{t[a]\}_{a \in \mathrm{cr}(a^*)}$ are all fixed, we consider $t[a^*]$ as ready, and submit question(a^*, $\mathrm{cr}(a^*)$, t) to the QA interface and fix $t[a^*]$ (line 15 - 16).

There may exist *deadlock* cases wherein two (or multiple) unfixed attributes appear in each others' correlation sets, while all the attribute values of their dependency sets are present (no missing values). In this case, we pick $t[a^*]$ with the least uncertainty in its correlation set (i.e., most of $\{t[a]\}_{a \in \mathrm{cr}(a^*)}$ are fixed) as the next value to be fixed (line 10 - 12).

Algorithm 2: Fixing Data Defects

```
   Input: tuple $t$, table pattern $\mathcal{G} = \{\mathrm{tp}(a), \mathrm{cr}(a)\}_a$
   Output: fixed tuple $t$"
1  $\mathcal{L}_{\text{queued}} \leftarrow \mathcal{A}$, $\mathcal{L}_{\text{fixed}} \leftarrow \emptyset$;
2  for $a^* \in \mathcal{L}_{queued}$ do
3      if $\mathrm{cr}(a^*) = \emptyset$ then
4          if $t[a^*]$ is missing then report $t[a^*]$ as "unfixable";
5          else add $a$ to $\mathcal{L}_{\text{fixed}}$;
6          remove $a$ from $\mathcal{L}_{\text{queued}}$;
7  while $\mathcal{L}_{queued} \neq \emptyset$ do
8      $\mathcal{L}_{\text{ready}} \leftarrow \emptyset$;
       // $t[a^*]$ is ready to be fixed
9      if $\exists a^* \in \mathcal{L}_{queued}, \mathrm{cr}(a^*) \bigcap \mathcal{L}_{fixed} = \mathrm{cr}(a^*)$ then add $a^*$ to
         $\mathcal{L}_{\text{ready}}$ ;
       // resolve deadlock
10     else if attributes $\{a\}$ are in deadlock then
11         pick $a^* = \arg\max_a \frac{|\mathrm{cr}(a) \bigcap \mathcal{L}_{\text{fixed}}|}{|\mathrm{cr}(a)|}$;
12         add $a^*$ to $\mathcal{L}_{\text{ready}}$;
13     if $\mathcal{L}_{ready} = \emptyset$ then break;
14     for $a^* \in \mathcal{L}_{ready}$ do
15         execute question($a^*$, $\mathrm{cr}(a^*)$, $t$) to fix $t[a^*]$;
16         move $a^*$ from $\mathcal{L}_{\text{queued}}$ to $\mathcal{L}_{\text{fixed}}$;
17 for $a \in \mathcal{L}_{queued}$ do report $t[a^*]$ as "unfixable";
```

In addition to suggesting possible fixes, DeepClean also presents the answer spans from Wikipedia as contextual information to the data analysts, who may then make final repair decisions.

6 EMPIRICAL STUDY

In this section, we empirically evaluate DeepClean using real-life tabular data. The experiments are designed along three dimensions: (i) the efficacy of pattern extraction, (ii) the efficacy of completion and repair, and (iii) case studies.

6.1 Experimental Setting

We first describe the setting of our experiments.

Free-text knowledge source. We use the SQL dump of the English Wikipedia as of 11/03/2017 as the free-text knowledge source, and adopt the DrQA model [6] as its question-answering (QA) interface, which together constitute DeepClean's backend, as shown in Figure 2.

Datasets. As currently there still lack benchmark datasets for data cleaning tasks, we collect four datasets from Web for evaluation: WikiTables,[2] DBPedia,[3] WebTables, and RelationalTables.

- WikiTables includes tables extracted from Wikipedia pages.
- DBPedia contains tables from the DBPedia knowledge base.

[2] http://downey-n1.cs.northwestern.edu/public/

[3] http://web.informatik.uni-mannheim.de/DBpediaAsTables/

dataset	# tables	# tuples	# attributes
WikiTables	19	668	54
DBPedia	10	417	24
WebTables	26	1,730	71
RelationalTables	2	9,949	5

Table 2. Statistics of the datasets used in the experiments.

- WebTables is a set of tables scraped from Web pages.
- RelationalTables contains two tables: Soccer[4] describes soccer players, their clubs, and nationalities; University[5] describes US universities and their addresses.

The statistics of the datasets are summarized in Table 2.

The attribute types (in terms of Wikipedia categories) and attribute correlations in all the tables are manually annotated, which we use as ground truth to measure DeepClean's performance.

All the algorithms are implemented in Python.

6.2 Experimental Results

Next we present the results of our empirical study. Due to space limitations. More details are referred to our technical report [36].

Pattern Extraction. Recall that in the pattern extraction stage, DeepClean automatically discovers the data semantics (including attribute types and correlations). For each attribute, we compare its attribute type and correlation set found by DeepClean against the human-annotated ground truth. As in realistic environments, we face possibly dirty data. To simulate such settings, we inject errors into the data by randomly selecting 10% cells in all the tables and modifying their values.

dataset	type	correlation
WikiTables	0.84	0.81
DBPedia	0.88	0.81
WebTables	0.77	1.0
RelationalTables	0.69	0.84

Table 3. DeepClean's accuracy of extracting attribute types and correlations (error rate = 10%).

Table 3 summarizes DeepClean's performance of extracting attribute types and correlations. It is noted that DeepClean attains reliable accuracy in terms of discovering types and correlations, with average accuracy above 0.6 and 0.84 across all the datasets. Especially in the WebTables dataset, DeepClean successfully discovers correct correlations for all the attributes.

dataset	type	correlation
WikiTables	0.72	0.83
DBPedia	0.76	0.92
WebTables	0.68	1.0
RelationalTables	0.60	0.84

Table 4. DeepClean's accuracy of extracting attribute types and correlations (error rate = 20%).

To assess the impact of data quality, we increase the error rate to 20% and report DeepClean's accuracy in Table 4. As expected, the accuracy of type extraction decreases modestly. Counterintuitively, the accuracy of correlation extraction increases slightly! This phenomenon may be explained as follows: DeepClean is designed to find highly correlated attributes; the random errors reduce spurious correlations and make true correlations more evident.

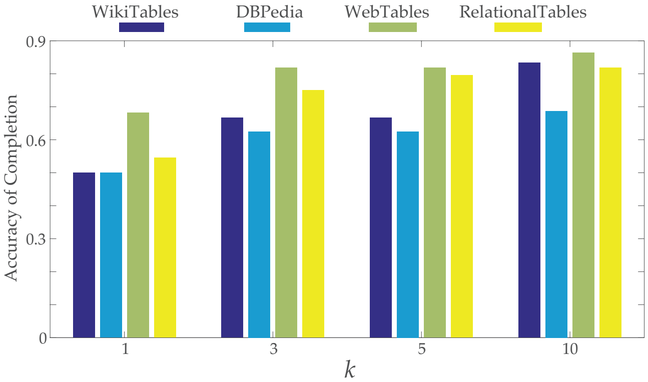


Figure 4: DeepClean's accuracy of filling missing values versus the setting of k.

Note that as we define its attribute types in terms of Wikipedia categories, it is difficult to directly compare DeepClean with other existing methods (e.g., [8, 16, 28]), which all require knowledge bases to define attribute types.

Completion and Repair. We differentiate two types of data cleaning tasks. In the completion task, DeepClean fills missing values using answered returned by the knowledge source. In the repair task, DeepClean first detects erroneous values and then replace them with returned answers. Next we report DeepClean's performance in both tasks.

[*Completion Tasks*] To evaluate DeepClean's performance in the completion task. We randomly select and delete 10% cells in all the tables; that is, each cell has 10% chance of being removed. We then estimate DeepClean's accuracy in terms of finding right answers for these missing values.

$$\text{accuracy} = \frac{\text{\# correctly filled values}}{\text{\# missing values}}$$

dataset	mean	std
WikiTables	0.71	± 0.11
DBPedia	0.68	± 0.15
WebTables	0.83	± 0.12
RelationalTables	0.88	± 0.01

Table 5. DeepClean's accuracy of filling missing values.

Table 5 summarizes the results. Observe that across all the datasets, DeepClean achieves accuracy above 0.68 in filling missing cells. Interestingly, DeepClean does not achieve the highest accuracy in the WikiTables dataset, which comprises tables extracted from Wikipedia pages. This may be explained by that DeepClean interacts with Wikipedia solely through its QA interface, which mainly relies on text content and ignores other structured information (e.g., tables) in Wikipedia pages.

In addition, we examine the impact of the parameter k on DeepClean's accuracy. Recall that k controls the number of candidate values suggested by DeepClean. We count a completion operation

[4] https://www.premierleague.com/

[5] https://ope.ed.gov/accreditation/

as correct only if the ground-truth value appears in the top k candidate values. Figure 4 shows DeepClean's completion accuracy as k varies from 1 to 10. It is noticed that in general DeepClean's performance is insensitive to the setting of k and the correct value appears in the top candidate values with high probability. In the following, we set $k = 5$ by default.

[*Repair Tasks*] As the repair task consists of two parts, detecting erroneous values and correcting them, in which the correction step is similar to the completion task. Therefore here we focus on DeepClean's performance of detecting erroneous values. To inject errors into the data, we randomly select and modify 10% cells in all the tables.

Recall that DeepClean verifies cell values via iterative question asking and reports suspicious cases if answers conflict with cell values. We measure DeepClean's accuracy in terms of precision and recall:

$$\text{precision} = \frac{\text{\# correct detection}}{\text{\# all detection}} \qquad \text{recall} = \frac{\text{\# correct detection}}{\text{\# all errors}}$$

dataset	precision	recall
WikiTables	0.82	0.82
DBPedia	0.47	0.73
WebTables	0.70	1.0
RelationalTables	0.60	1.0

Table 6. DeepClean**'s accuracy of detecting erroneous cases.**

dataset	precision	recall
WikiTables	1.0	0.30
WebTables	1.0	0.46

Table 7. Katara**'s accuracy of repairing erroneous cases.**

Table 6 summarizes DeepClean's accuracy of detecting erroneous values. It is observed that DeepClean enjoys high detection accuracy in most of the cases, which especially manifests in its recall. For comparison purpose, Table 7 lists the performance of Katara [8],[6] a knowledge base-assisted data cleaning method, on similar datasets. Interestingly, the comparison shows the strengths and limitations of the two classes of approaches. Thanks to the broader coverage of free-text encyclopedias, DeepClean is able to detect a larger number of erroneous cases. Meanwhile, knowledge bases offer more strict knowledge representation, which enables Katara to pinpoint a subset of erroneous cases with higher precision. This observation implies the possible synergistic effects of integrating DeepClean with knowledge base-assisted methods to deliver more effective data cleaning tools.

Case Studies. Next we compare two concrete cases, in one case DeepClean achieves high repair accuracy while in the other it does not perform well, and discuss its strengths and limitations.

We pick two tables from the WebTables dataset, Table 1 - American Films and Actresses and Table 2 - National Film Awards, in which DeepClean's accuracy of filling missing values is 0.33 and 0.81 respectively. Figure 5 shows the table patterns found by DeepClean for the two tables.

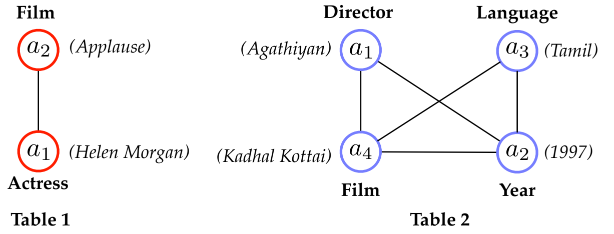


Figure 5: Patterns of two sample tables (the values in the parenthesis represent one data instance).

Notice that Table 1 describes the simple relationships between actresses and their films. However, since such relationships are often non-bijectional (often n-to-n), it is challenging for DeepClean to extract relevant answers to verify given data. In contrast, Table 2 describes the complicated relationships of film directors, award winning films, award years, and film languages. For example, the award year is correlated with both the director and the film, while the director or the film alone is insufficient to determine the award year. Given such multiple constraints, DeepClean is able to perform effective question asking by using such constraints as rich contexts.

We can therefore conclude that DeepClean is especially suitable for dataset with complicated inter-attribute dependencies.

7 RELATED WORK

In this section, we review three categories of related work, namely, data cleaning, machine comprehension of text, and Wikipedia-based knowledge discovery.

In response to the practical need of automated and dependable data cleaning, a plethora of solutions have been proposed. One line of work solely relies on internal information of the datasets, such as integrity constraints [4, 7, 12, 21, 25] and statistics [18, 24, 33]. Despite their generality, such best-effort solutions often fail to precisely identify and correct errors, due to the limitations of available information. Another line of work exploits external information, such as master data [11], domain expertise [23, 34], crowdsourcing [29], and knowledge bases [8]. Some recent work [35] further considers the populating tabular data with the assistance of such knowledge sources. However, such resources are often scarce, expensive to employ, or limited by incompleteness and fixed schema. In comparison, free-text knowledge sources are much more abundant, expressive, and updated, making them ideal backends for data cleaning tasks.

Machine comprehension of text is the problem of answering questions after reading short texts. Thanks to advances in deep neural network models (e.g., attention- and memory-augmented networks [3, 26]) and newly available datasets (e.g., CNN/Daily Mail [13] and SQuAD [22]), machine comprehension models have achieved capabilities comparable to humans on some QA tasks [30], opening the opportunity of building user-friendly QA interfaces for free-text knowledge sources [6, 10, 31]. To our best knowledge, this work is the first to exploit such QA interfaces for data cleaning.

Finally, besides serving as knowledge sources for QA, Wikipedia is also widely used in other knowledge discovery tasks, such as document clustering [15], named entity disambiguation [9], and information diffusion tracking [27]. However, to our best knowledge, this work is among the first to perform high-quality data cleaning based on Wikipedia through its question-answering interface.

[6] As Katara requires crowdsourcing, Table 7 copies the results reported in [8].

8 CONCLUSION AND FUTURE WORK

We propose DeepClean, the first end-to-end data cleaning framework powered by free-text knowledge sources. Designed for a cold-start setting wherein the data semantics is unknown a priori, DeepClean automatically discovers the attribute types and correlations by bridging the data with the knowledge source through its question-answering interface. Each data tuple is validated and repaired by issuing a minimal set of questions to the knowledge source and checking the returned answers against the tuple values. Extensive experiments on real datasets demonstrate that DeepClean is applicable to a range of domains and can effectively repair a variety of data defects.

ACKNOWLEDGMENTS

We would like to thank anonymous reviewers for insightful comments. This material is based upon work supported by the National Science Foundation under Grant No. 1566526 and 1718787. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation

REFERENCES

[1] S. Fissaha Adafre, V. Jijkoun, and M. de Rijke. 2007. Fact Discovery in Wikipedia. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Web Intelligence.*

[2] Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging Linguistic Structure For Open Domain Information Extraction. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL '15).*

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR '15).*

[4] Philip Bohannon, Wenfei Fan, Michael Flaster, and Rajeev Rastogi. 2005. A Cost-based Model and Effective Heuristic for Repairing Constraints by Value Modification. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD '05).*

[5] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08).*

[6] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL '17).*

[7] Fei Chiang and Renee J. Miller. 2011. A Unified Model for Data and Constraint Repair. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering (ICDE '11).*

[8] Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. KATARA: A Data Cleaning System Powered by Knowledge Bases and Crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15).*

[9] Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

[10] Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu. 2017. Attention-over-Attention Neural Networks for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL '17).*

[11] Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Wenyuan Yu. 2012. Towards Certain Fixes with Editing Rules and Master Data. *The VLDB Journal* 21, 2 (2012), 213–238.

[12] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2013. The LLUNATIC Data-cleaning Framework. *Proc. VLDB Endow.* 6, 9 (July 2013), 625–636.

[13] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15).*

[14] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. *Artif. Intell.* 194 (2013), 28–61.

[15] Xiaohua Hu, Xiaodan Zhang, Caimei Lu, E. K. Park, and Xiaohua Zhou. 2009. Exploiting Wikipedia As External Knowledge for Document Clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09).*

[16] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and Searching Web Tables Using Entities, Types and Relationships. *Proc. VLDB Endow.* 3, 1-2 (2010), 1338–1347.

[17] Steve Lohr. 2014. For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights. https://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html. (2014).

[18] Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. 2010. ERACER: A Database Approach for Statistical Inference and Data Cleaning. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10).*

[19] Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. 2011. DBpedia SPARQL Benchmark: Performance Assessment with Real Queries on Real Data. In *Proceedings of the 10th International Conference on The Semantic Web - Volume Part I (ISWC '11).*

[20] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes* 30, 1 (January 2007), 3–26.

[21] Paolo Papotti, Xu Chu, and Ihab F. Ilyas. 2013. Holistic Data Cleaning: Putting Violations into Context. In *Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE 2013) (ICDE '13).*

[22] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *ArXiv e-prints* (2016).

[23] Vijayshankar Raman and Joseph M. Hellerstein. 2001. Potter's Wheel: An Interactive Data Cleaning System. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01).*

[24] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proc. VLDB Endow.* 10, 11 (2017), 1190–1201.

[25] Shaoxu Song, Hong Cheng, Jeffrey Xu Yu, and Lei Chen. 2014. Repairing Vertex Labels Under Neighborhood Constraints. *Proc. VLDB Endow.* 7, 11 (July 2014), 987–998.

[26] Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *Advances in Neural Information Processing Systems 28.*

[27] Jintao Tang, Ting Wang, Qin Lu, Ji Wang, and Wenjie Li. 2011. A Wikipedia Based Semantic Graph Model for Topic Tracking in Blogsphere. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11).*

[28] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering Semantics of Tables on the Web. *Proc. VLDB Endow.* 4, 9 (2011), 528–538.

[29] Jiannan Wang and Nan Tang. 2014. Towards Dependable Data Repairing with Fixing Rules. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14).*

[30] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated Self-Matching Networks for Reading Comprehension and Question Answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL '17).*

[31] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making Neural QA as Simple as Possible but not Simpler. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL '17).*

[32] K. Xu, S. Reddy, Y. Feng, S. Huang, and D. Zhao. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL '16).*

[33] Mohamed Yakout, Laure Berti-Équille, and Ahmed K. Elmagarmid. 2013. Don'T Be SCAREd: Use SCalable Automatic REpairing with Maximal Likelihood and Bounded Changes. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13).*

[34] Mohamed Yakout, Ahmed K. Elmagarmid, Jennifer Neville, Mourad Ouzzani, and Ihab F. Ilyas. 2011. Guided Data Repair. *Proc. VLDB Endow.* 4, 5 (2011), 279–289.

[35] Shuo Zhang and Krisztian Balog. 2017. EntiTables: Smart Assistance for Entity-Focused Tables. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17).*

[36] X. Zhang, Y. J, C. Nguyen, and T. Wang. 2018. Data Cleaning by Question Asking (Technical Report). *ArXiv e-prints* (2018).