Context Retrieval for Web Tables

Hong Wang*, Anqi Liu*, Jing Wang*, Brian D. Ziebart*, Clement T. Yu*, Warren Shen†

*Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
{hwang207, aliu33, jwang69, bziebart, cyu}@uic.edu

†Cohesity
451 El Camino Real
Santa Clara, CA 95050
warren@cohesity.com

ABSTRACT

Many modern knowledge bases are built by extracting information from millions of web pages. Though existing extraction methods primarily focus on web pages' main text, a huge amount of information is embedded within other web structures, such as web tables. Previous studies have shown that linking web page tables and textual context is beneficial for extracting more information from web pages. However, using the text surrounding each table without carefully assessing its relevance introduces noise in the extracted information, degrading its accuracy. To the best of our knowledge, we provide the first systematic study of the problem of table-related context retrieval: given a table and the sentences within the same web page, determine for each sentence whether it is relevant to the table. We define the concept of relevance and introduce a Table-Related Context Retrieval system (TRCR) in this paper. We experiment with different machine learning algorithms, including a recently developed algorithm that is robust to biases in the training data, and show that our system retrieves table-related context with F1=0.735.[1]

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Web tables; context retrieval; covariate shift

1. INTRODUCTION

The construction of modern knowledge bases (KBs) has gained popularity in recent years. Most KBs [29, 7, 2, 21, 27, 37, 32] are built by extracting information from millions of texts. In addition to web text resources, there is also a huge amount of information embedded within other web structures, such as web tables.

Several studies [4, 5, 6] have been conducted to extract information in web tables. Dong et al. [9] extract the entities and relations (defined in a fixed ontology) from text and tables separately. Govindaraju et al. [11] claim that there are many cues buried in the surrounding text that enable us to understand the table. Compared with modeling the text and table separately, jointly modeling both of them extracts more relations. However, Limaye et al. [19] show that simply relating surrounding context with the table also introduces unrelated information to table information extraction. Gupta et al. [12] mention that 41% of errors in their algorithm for recovering web tables' semantics are caused by unrelated information in the text surrounding tables, and they suggest using table-related text instead.

Motivated by these previous observations, and Knowledge-based applications, such as Semantic Search and Question Answering (QA) (Section 2), we develop a Table-Related Context Retrieval system (TRCR) for identifying texts that are relevant to a table. Our approach differs from related work (Section 3) in problem formulation and the supervised machine learning approach we employ. We define the concept of "relevance" between web table and sentences within the same page in Section 4 and our method for identifying the relevant sentences in Section 5. This includes a description of the features we engineered for the task and machine learning methods that address the situation that annotated training data may not represent the test data well. We believe this latter concern—source sample bias—is an important but overlooked issue for data that is annotated at the document level, producing many training examples for a small number of documents. We introduce our data sets and demonstrate the benefits of our approach in Section 6. We conclude by discussing future work in Section 7.

2. MOTIVATION AND CONTRIBUTIONS

In this section, we use two examples to demonstrate the motivations that lead to our study. The first example demonstrates that retrieving table-related context improves the coverage of QA. Consider sending the following question to Google:

$Q1$: *"How many episodes did Constance Zimmer act in Boston Legal?"*.

From the returned snippet shown in Figure 1, we find no certain answer. Entering the first retrieved web page `http://en.wikipedia.org/wiki/Constance_Zimmer`, by in-

[1]Part of this work was done during the first author took his summer internship with Google Inc., under the supervision of the last author in year 2013.

ICTIR'15, September 27–30, Northampton, MA, USA.
 ISBN 978-1-4503-3833-2/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2808194.2809453.

Figure 1: Snippets from Google for "How many episodes did Constance Zimmer act in Boston Legal?"

specting text only, we find the most related sentence to $Q1$:

$S1$: *"She joined the cast of Boston Legal, where she played associate attorney Claire Simms on..."*

where the triples:

(she, joined the cast of, Boston Legal) and

(she, played, associate attorney Claire Simms),

can be extracted by Open Information Extraction (Open IE) tools, like ReVerb [10]. By utilizing coreference resolvers (e.g. Stanford Deterministic Coreference Resolution System [25, 18]), *"she"* is replaced with *"Constance Zimmer"*. Unfortunately, this information is still not sufficient to answer the question.

By inspecting the table (Figure 2) in the same page alone, we find a row that contains the words *"Boston Legal"* and *"Claire Simms"*. However without matching *"Constance Zimmer"* with *"Claire Simms"*, even though *"23 episodes"* in the last cell of the row is the correct answer, the question is still not answerable.

2006–2007	*Boston Legal*	Claire Simms	23 episodes

Figure 2: Table from Wikipedia page "Constance Zimmer".

By realizing $S1$ is related to the table, the following information can be linked together. The QA system can then utilize such linked information to infer the answer for $Q1$.

(Constance Zimmer, joined the cast of, Boston Legal)

(Constance Zimmer, played, associate attorney Claire Simms)

(Boston Legal, Claire Simms, 23 episodes)

This example shows that using text information jointly with a table can provide information beyond that of the sentence alone and that of the table alone.

The second example shows that retrieving only table-related context can avoid irrelevant information and improve the QA accuracy. Consider the following the question:

$Q2$: *"Which mines were closed in Yellowknife?"*

Google again returns snippets instead of any certain answer (see Figure 3). Notice that the 4th returned snippet *"The Diavik Diamond Mine"* is actually incorrect since this mine is still operating, even it appears in the corresponding Wikipedia page `http://en.wikipedia.org/wiki/Yellowknife`:

$S2$: *"A second mine, Diavik Diamond Mine, began production in 2003."*

$S3$: *"The following is a list of the major mines, all of which are now closed."*

$S2$ is irrelevant to the table on the same page (Figure 4), since the table is about the mines are now closed. On the contrary, $S3$ is not only relevant to that table, but also matches $Q2$. From this demonstration, we see that by combining the table with related context (e.g. $S3$), while eliminating irrelevant information (e.g. $S2$), it is possible to obtain highly accurate answers to the questions.

Figure 3: Snippets from Google for "Which mines were closed in Yellowknife?" *(tailored to fit in the page)*

Mine	Years of Operation	Minerals Mined
Con Mine (includes Rycon)	1938–2003	gold
Giant Mine	1948–2004	gold

Figure 4: Table from Wikipedia page "Yellowknife".

In this paper, we provide an in-depth study of the table-related context retrieval problem: given a table and the sentences within the same web page, determine for each sentence whether it is relevant to the table. The contributions of this paper are fourfold:

1) To the best of our knowledge, we are the first to define and systematically study the problem of retrieving table-related context on web pages. We exploit six types of relatedness between the sentence and the table to define the notion of relevance.

2) We propose the Table-Related Context Retrieval system (TRCR) that solves the table-related sentence retrieval problem. Our approach employs 61 features to characterize text-table relationships, including page structure, token matching, and semantic analysis.

3) We provide a new set of data, where each instance is a table-sentence pair. Each pair is labeled as RELEVANT or IRRELEVANT by different annotators with stable agreement. It provides a gold standard that other researchers can perform experiments on[2].

4) We experiment with different machine learning algorithms, including a recently developed algorithm that is robust to biases in the training data, and show that our system retrieves table-related context with F1=0.735.

3. RELATED WORK

Google's WebTables [4, 5, 6] pioneered the study of extracting and leveraging web tables that contain billions of instances of high-quality relational information. Their current work mostly focuses on extracting relational data from HTML tables, without connecting tables with their related context.

Knowledge Vault [9] extracts the entities and relations (defined in a fixed ontology) from texts and tables sepa-

[2]Code and data can be found at `https://code.google.com/p/uic-cs-dbis/`.

rately. Due to the limited textual content in the tables, the relations between columns in tables cannot be captured by standard schema matching methods [30], and only a small amount of information is extracted from the tables.

A recent study [11] has found that the surrounding text of a table can help improve human recall capability by more than 60% for humans trying to understand table content. Motivated by this, their model for relation extraction from tables achieves an F1 score twice as high as the performance of either pure-table or pure-text systems. However in that work, the entire document text is considered to be context related to the table and is used to extract relations.

Limaye et al. [19] use probabilistic graphical models to annotate web table columns, cells, and pairs of columns with types, entities (persons, organizations, locations, etc.), and binary relations from the YAGO catalog [29] respectively. The table content (headers and data rows), and also some amount of textual context around table are used in matching. However, the authors of that work only mention that they "capture some amount of textual context around tables" and point out that those textual contexts contain irrelevant information; neither a detailed algorithm nor a measurement of relevance between the captured surrounding sentence and the table is given.

Biperpedia is an ontology built by Google [12]. It is used to improve the performance of recovering the semantics of web tables by mapping columns to Biperpedia attributes. Because web tables have no schema, the surrounding text is particularly important in the mapping algorithm. Similar to the work using probabilistic graphical models [19], the authors also indicate that the surrounding text often also contains irrelevant information, and the most frequent cause of error in their algorithm is due to this irrelevant information.

System WWT [23] takes a query with a set of column keywords, extracts relevant columns from KB of web tables, consolidates them, and returns a single structured table. Headers, data cells and the surrounding context of the table are used to match query keywords in WWT. It considers "any text node x that is a sibling of a node on the path from T to the root of d" as the related context, where T is the node in DOM tree d of the web page that contains the table. However, in that paper, relatedness is not explicitly given, and their approach seems[3] to extract context by structure only. Microsoft's InfoGather [34] measures the relatedness between a table and its surrounding text by tf-idf cosine similarity functions. In contrast to their work, our system utilizes structural, content, and semantic features.

Despite the common agreement across multiple studies that related text is useful for understanding the table and extracting relations, no systematic study on web table related context retrieval has been performed to our knowledge.

4. DEFINITION OF RELEVANCE

In this work, we study the problem of retrieving table-related context from HTML web pages. The context is relevant to a table if it is related to the contents of the table and it could provide information beyond that of the sentence alone and that of the table alone. Given a web table as input, our proposed Table-Related Context Retrieval system (TRCR) retrieves the context on the same web page that is relevant to the table. Similar to a previous study [31], we limit our consideration to horizontal HTML tables (i.e., table headers are aligned in the first row(s)). Since a sentence is a grammatical unit of one or more words that expresses an independent statement, question, request, command, exclamation[4], and most existing Open IE systems like TextRunner [3] and ReVerb [10] extract relations from English sentences, we define the concept of "context" as sentences in the same HTML page as the table of interest. Based on our observation, six types of relatedness between the sentence and the table are exploited to define the notion of relevance as follows. We provide example from Wikipedia of each type. Note that in this study, we focus on using this six types of relatedness to help us define the relevance, rather than differentiating these six types, which is left as one of our future works.

1) Table summary
A sentence that summarizes the table's content and/or what the table is about. As we demonstrated in the second example in Section 2, $S3$ is in such relevance. By combining the sentence and table, the irrelevant information (i.e. $S2$) could be filtered out like we showed in that example.

2) Header description
A sentence that describes what a header of a table column is, and/or any attribute that the header has. For example, the sentence *"Model organisms are in vivo models and are widely used to research human disease when..."* in page `http://en.wikipedia.org/wiki/Model_organism`, describes an attribute of the first column header (i.e. "Organism") of the following table (Figure 5).

Organism	Genome Sequenced
Prokaryote	
Escherichia coli	Yes
Eukaryote, unicellular	
Dictyostelium discoideum	Yes
Saccharomyces cerevisiae	Yes

Figure 5: Table from Wikipedia page "Model organism".

3) Header relation
A sentence that expresses a relation involving at least two column headers of the table. For example, in `http://en.wikipedia.org/wiki/List_of_Eurovision_Song_Contest_winners`, the sentence *"The country awarded the most points is declared the winner"* expresses a relation between two headers: "winner" and "points" in the table in Figure 6.

Year	Date	Host City	Winner	Song
Performer	Writers	Points	Margin	Runner-up

Figure 6: Table from Wikipedia page "List of Eurovision Song Contest winners".

4) Data relation
A sentence that expresses a relation among two or more data cells (usually in one table row, but not necessarily so)

[3]No precise algorithm is given in that paper.

[4]"Sentence" - Definitions from Dictionary.com (`http://dictionary.reference.com/browse/sentence`)

in web table. For example, in `http://en.wikipedia.org/wiki/Erdos-Bacon_number`, the sentence *"Astronomer Carl Sagan has an Erdős number of no more than 4 and a Bacon number of 2, for a total of 6"* expresses a relation involving all four data cells within a row in the table in Figure 7.

Name	Erdős number	Bacon number	Erdős–Bacon number
Carl Sagan	4[6]	2[b][5]	6

Figure 7: Table from Wikipedia page "Erdős-Bacon number".

5) Header-data relation
A sentence that expresses a relation among two or more columns of the web table, including at least one column header, and one data cell. For example, for the same page and table as the example above (Figure 7), the sentence *"Notable scientists with defined Erdős-Bacon numbers include popular astronomer Carl Sagan"* expresses a relation between the header "Erdős-Bacon numbers" and the data cell "Carl Sagan".

6) Page subject-data relation
There's another type of relevance where the whole web page talks about someone or something (we call it the subject of the page), while the content of the web table in that page is all about aspects of that subject. A sentence talks about the page subject's certain attributes which are also expressed in the table. For example, as we already seen in the first example in Section 2, the subject of the web page is "Constance Zimmer", and the sentence *S1* is related to the table in this type of relevance. With the combination of the sentence and the table, we could infer the information as we demonstrated.

5. TABLE-RELATED CONTEXT RETRIEVAL SYSTEM (TRCR)

5.1 Overview

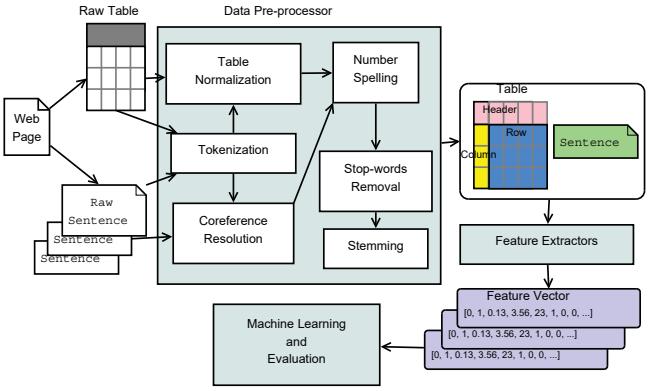

Figure 8: Architecture of TRCR.

Our proposed TRCR system consists of three main components (see Figure 8):
• Data pre-processor: all the sentences are read from our labeled data set, together with table content. Text from sentences and tables is processed using standard IR techniques, which we will discuss in detail in the following section.
• Feature extractors: from the pre-processed data, we extract 61 features, 33 of them are boolean-valued, the rest are numerical features.
• Machine learning algorithms and evaluators: extracted features from previous component are sent to different machine learning algorithms. Models are trained on a training data set, and evaluated on a test data set.
In the following sections, we first introduce our data preprocessing methods. Next, we provide a detailed description of the features we engineer for this task. Finally, we introduce several algorithms, together with the motivation and the merit of considering source data sample bias.

5.2 Data pre-processing

Tokenization, stop words removal, stemming
Like other standard Information Retrieval systems, we first tokenize sentences and table content. We use Apache Lucene [13] in our TRCR. Stop words are removed from tokens. Tokens are then stemmed using the Porter Stemmer algorithm [24]. Numbers are also normalized during tokenization, e.g. "12,000,000" becomes "12000000". Diacritics are removed from non-English letters, e.g. "Yves Allégret" becomes "Yves Allegret".

Number spelling
Tables often contain numeric tokens, but a relevant sentence contains the spelled forms, or vice versa. So for each numeric token, we spell it out in English using ICU4J[5]. For example, on page `http://en.wikipedia.org/wiki/World_Series_of_Poker`, the sentence *"Gold pocketed US$12 million for his victory"* has a numeric token *"12"*, which is spelled out as *"twelve"*; while in the table there's a data cell *"12000000"* that is also spelled out in English as *"twelve million"*. The spelled out English tokens are added as synonyms of the original numeric token for further text matching. In addition, "2" and "3" are also added and spelled as synonyms of "twice", and "thrice" respectively, since they are very common replacements of "two times" and "three times."

Coreference resolution
We also use a coreference resolver [25, 18] on each entire web page to resolve the coreferences of third-person pronouns in sentences. For example, on the page `http://en.wikipedia.org/wiki/Rani_Mukerji`, *"she"* and *"her"* in the sentence *"At the age of fourteen, she was cast by her father for a supporting role in his Bengali film Biyer Phool (1992)"* are replaced by *"Rani Mukerji"* through coreference resolution. Because the models of the coreference resolver available online are for the CoNLL-2011 shared task, which is different from our Wikipedia data set, its accuracy is far from perfect[6]. We employ some modifications based on the original resolution results to accommodate our data set and increase its accuracy. The input of our modified coreference resolution algorithm are: token t; sentence S which token t belongs to; coreferences C of token t that are resolved by the original resolver; the most representative mention $r \in C$ that the original resolver selected; heading h of the section that sentence S belongs to; and page title p. The algorithm's outputs are tokens R that are considered as replacements of token t. It resolves only third person pronouns, and processes in

[5]International Components for Unicode, http://site.icu-project.org/
[6]http://nlp.stanford.edu/software/dcoref.shtml

steps. The algorithm first checks the most representative mention but with an additional constraint that requires this representative mention to appear in the current sentence or to have already appeared in an earlier sentence. If these conditions are both satisfied, representative mention is returned as the replacement of input token t. If not, our algorithm selects the longest coreference that appears in either page title or sentence section heading as the replacement. If there's still no replacement found (i.e., each coreference $c \in C$ is neither recognized as representative by the original resolver that appears in/before sentence S, nor appears in the title or sentence section heading), the algorithm checks three sentences ahead together with the current sentence. The closest coreference to token t that has all tokens with their initial letters capitalized is considered as the replacement. Details of this modified coreference resolution algorithm in TRCR are listed in Algorithm 1. Its accuracy on our training data set is 82.33%.

Algorithm 1 Coreferences Resolution
INPUT: t, S, C, r, h, p; **OUTPUT**: R

```
1: // 3rd person pronouns
2: 3PRP ← {"he", "him", "his", "she", "her", "hers", "it", "its",
   "they", "them", "their", "theirs"};
3: // 1st and 2nd person pronouns
4: 12PRP ← {"i", "me", "my", "mine", "we", "us", "our", "ours",
   "you", "your", "yours"};
5: if C = Φ || t ∉ 3PRP then return Φ;
6: R ← Φ;
7: S_ ← any_sentence_before(S);
8: for all c ∈ C do
9:     if t ∉ 3PRP && t ∉ 2PRP && c = r
10:      && (c ∈ S || c ∈ S_) then
11:        R ← R + c;
12: if R = Φ // no replacement retrieved yet then
13:     // longest coreference first
14:     sort_in_descending_order_of_length(C);
15:     for all c ∈ C do
16:         if t ∉ 3PRP && t ∉ 2PRP
17:           && (c ∈ p || c ∈ p) then
18:             R ← R + c; break;
19: if R = Φ // no replacement retrieved yet then
20:     // closest coreference first
21:     sort_in_descending_order_of_distance_to_t(C);
22:     S_1−2−3 ← at_most_3_sentences_away_before(S);
23:     for all c ∈ C do
24:         if t ∉ 3PRP && t ∉ 2PRP
25:           && has_all_tokens_initial_capitalized(c)
26:           &&(c ∈ S || c ∈ S_1−2−3) then
27:             R ← R + c; break;
28: return R;
```

Web table normalization
Some table columns may contain "yes" or "no" terms only, like the table shown in Figure 5. We replace "yes" with the column header ("Genome Sequenced" in this example). Each "no" is ignored since cells with "no" in them are supposed to be irrelevant to the header.
A table may also contain "year" columns in which consecutive years are represented in the pattern "$xxxx - yyyy$", where "$xxxx$" is the start year, and "$yyyy$" is the end year, like those shown in Figure 4. We check if the header of the table contains the token "year" or "years". If the header contains any of them, we check each data cell content in that column to see if it contains text in the pattern "$xxxx-yyyy$" (white-spaces within this pattern are optional), and "$xxxx$", "$yyyy$" are numeric tokens where $xxxx < yyyy$. If it does, we consider the cell to contain consecutive years, and "refill" the cell data by adding all years between $xxxx$ and $yyyy$ into it. For example, *"1941-1943, 1947-1949"* becomes *"1941, 1942, 1943, 1947 1948, 1949."*

After pre-processing, each sentence in the web page is stored and indexed by its section number, paragraph number and sentence number with respect to its position in the original web page (a section is considered a sequence of paragraphs led by a heading, except the first section). The web table is stored separately from the page's textual content, and is indexed by its section number.

5.3 Feature engineering

We have 61 features in two categories. One is from the sentence alone. The others are extracted from both the sentence and the table. In the rest of this section, we describe the features of each category and an intuition for their usefulness in detail.
Sentence features (18 features)
Sentence position. We use nine boolean features to capture the position information about each sentence, i.e., whether the sentence is the first sentence, the last sentence, or neither first nor last of: the entire page, a section, or a paragraph. We observe the phenomena that the first sentence in a web page often provides a summarization of the topic of the page; the first sentence in same section as the table often summarizes the content of the table, etc.
Sentence-title overlapping. The title of the web page can be considered the subject of the page, and may have some degree of relevance to the table. So we extract two features to check whether or not any or all of the tokens in the page title exist in the sentence.
Sentence-title semantic overlapping. In a sentence, there are some tokens that are more semantically indicative than others. In order to extract such tokens, we integrate ReVerb [10] to find semantic binary relationships. Each relationship is in the format of a triple (argument1, relation, argument2), where each component of the triple is a phrase that is extracted from the given sentence. We use a boolean feature for each component to indicate if it overlaps with the title.
"Table" or "list". For "table summary" relevance, we observe that the sentence usually contains the term "table" or "list", like the sentence "The following table presents..." So we create a boolean feature to check whether the sentence contains "table" or "list."
Sentence sentiment. We observe that the distributions of sentiment polarity (i.e., positive, negative, neutral) are different between RELEVANT and IRRELEVANT sentences. Positive or negative sentences are more likely to be RELEVANT when compared with neutral sentences; whereas neutral sentences have the highest chance of being IRRELEVANT. In this work, we integrate the Recursive Neural Tensor Networks (RNTN) method [28] to analyze the sentiment for each sentence. We use three boolean features to represent the possible polarities.

Table-sentence features (43 features)
Table-sentence distance. Intuitively, when a sentence is closer to the table, it is more likely to be relevant. We use a boolean feature to indicate whether the sentence and the table are in the same section. Across different sections, we observe that a sentence before the table has a higher chance of being relevant than a sentence after the table. In order to capture this insight, we create two additional numeric features to measure the section-distance in the following way:

suppose the total number of sections in the web page is N, the sentence is in section i, the table is in section j, then:

$$SD\%^- = \begin{cases} (j-i)/N & \text{if } i <= j \\ 0 & \text{otherwise,} \end{cases} \quad SD\%^+ = \begin{cases} (i-j)/N & \text{if } i >= j \\ 0 & \text{otherwise,} \end{cases}$$

where $SD\%^-$ represents the relative section-distance that the sentence and the table are apart, if the sentence is located before the table. Similarly, $SD\%^+$ represents the relative section-distance if the sentence is located after the table.

Header description. Sentences may use an equals sign ("=") to explain a table header's meaning, especially for abbreviations or acronyms, like "HR = Home runs". Also, sentences that follow the pattern *"<header> <be>"* (where "<be>" is a conjugation of the verb "to be", e.g., "is", "are", "was", "were") are likely to be a description of the header. We use boolean features to indicate whether the sentence contains such patterns.

Structure-indicative numbers. If a sentence talks about the table's topic, it is possible that the sentence provides a count of some data in the table. For example, in the sentence "Major stock exchanges (top 20 by market capitalization)..." from `http://en.wikipedia.org/wiki/List_of_stock_exchanges`, the number "20" is exactly the number of rows in the corresponding table. So, in order to capture such patterns, we create a boolean feature to check whether any number in the sentence matches the total number of data rows. Furthermore, a relevant sentence may count the number of occurrences of some data cell's content. Consider the sentence "The country with the highest number of wins is Ireland, with seven" in the page `http://en.wikipedia.org/wiki/List_of_Eurovision_Song_Contest_winners`. Information expressed by it can be found from the "winner" column of the table, within which "Ireland" occurs exactly seven times. One boolean feature is built to capture such cases.

Sentence-header overlapping. A header is considered as an important clue of the information in the table column. If the sentence shares common tokens with the header, it is likely to be of header-related relevance. Suppose the web table has C columns, each header of column is h_j, s is the sentence. We define five features as follows:

$$\#H_{any} = \sum_{j=1}^{C} I_{any}(h_j, s), \qquad \#H_{all} = \sum_{j=1}^{C} I_{all}(h_j, s),$$
$$H_{any}\% = \frac{\#H_{any}}{C}, \qquad H_{all}\% = \frac{\#H_{all}}{C},$$
$$SH\% = \frac{|s \bigcap_{j=1}^{C} h_j|}{|s|},$$

where $|s \bigcap_{j=1}^{C} h_j|$ is the number of common tokens in both sentence s and header h_j, $|s|$ is the number of tokens in sentence s, and:

$$I_{any}(h_j, s) = \begin{cases} 1 & \text{if any token in } h_j \text{ exists in s} \\ 0 & \text{otherwise,} \end{cases}$$

$$I_{all}(h_j, s) = \begin{cases} 1 & \text{if all tokens in } h_j \text{ exist in s} \\ 0 & \text{otherwise.} \end{cases}$$

Sentence-table overlapping. We use a set of features to measure the overlap between the sentence and the table row or column. In addition to a count of the common tokens, the alphabetic and numeric tokens are also analyzed separately. We describe some of them formally as follows: assume R is the total number of rows in a table, C is the number of columns, and c_{ij} represents the content of the data cell in row i and column j:

$$CR_{any} = \max_{i=1\ldots R} \sum_{j=1}^{C} I_{any}(c_{ij}, s), \qquad CR_{all} = \max_{i=1\ldots R} \sum_{j=1}^{C} I_{all}(c_{ij}, s),$$
$$CC_{any} = \max_{j=1\ldots C} \sum_{i=1}^{R} I_{any}(c_{ij}, s), \qquad CC_{all} = \max_{j=1\ldots C} \sum_{i=1}^{R} I_{all}(c_{ij}, s),$$
$$CR_{any}\% = \frac{CR_{any}}{C}, \qquad CR_{all}\% = \frac{CR_{all}}{C},$$
$$SR\% = \max_{i=1\ldots R} \frac{|s \bigcap_{j=1}^{C} c_{ij}|}{|s|}.$$

Sentence-header/table semantic overlapping. Similarly, we use ReVerb to extract relational triples from the sentence. Then, header tokens and table cell tokens are checked against each component of them to determine whether they share any or all of the tokens in common, generating six boolean features.

Sentence-heading overlapping. Usually, a table section's heading, as well as the table caption (if it exists), summarize the table content. When the sentence shares tokens with them, it has a higher probability of being relevant to the table. Hence, boolean features are built to indicate whether the overlaps exist.

5.4 Machine learning algorithms

We construct the TRCR system using supervised machine learning methods to estimate the probability of a sentence being relevant to a table given the features characterizing the sentence-table relationships. We denote this estimated distribution as $\hat{P}(y|x)$. The vast majority of supervised machine learning methods assume that data available for training the method is representative of (i.e., drawn from the same distribution as) data that the method will encounter at "test time." When data labels are provided in units of entire documents, and annotation is relatively expensive, a disproportionate amount of training data is produced from a small set of documents, creating non-representative training data sets. In addition to evaluating the performance of prevalent machine learning algorithms that ignore possible non-representative training data, including Decision Trees, k-Nearest Neighbors, Naïve Bayes, Support Vector Machine (SVM), and Logistic Regression, we evaluate two methods that take training set bias into account. These methods consider the special case of covariate shift [26] sample selection bias in which the conditional sentence relevance given input features is shared between training and test data sets, but the distribution of the input features differs between data sets: $P_{\text{train}}(x)P(y|x) \neq P_{\text{test}}(x)P(y|x)$.

5.4.1 Sample reweighted method

We consider a traditional approach for learning under covariate shift that uses importance weighting of the empirical loss function for the training distribution to approximate the expected loss for the test data distribution [36, 14, 8]. We call this the Sample Reweighted method (SR) in this paper. From the transfer learning perspective, the sample reweighted approach is a case of transductive transfer learning [22] and has been used in tasks including name entity recognition [1, 15], text categorization, and sentiment classification [33] in natural language processing. Model param-

eters θ using this method are chosen by:

$$\min_{\theta} \mathbb{E}_{\tilde{P}_{\text{train}}(x)P(y|x)} \left[\frac{P_{\text{test}}(X)}{P_{\text{train}}(X)} \text{loss}(\hat{P}_{\theta}(Y|X), Y) \right].$$

Conceptually, this produces parameter estimates that minimize predictive loss on sample data from $\tilde{P}_{\text{train}}(x)$ that is more representative of the test data set. Note that labels from the test data set are not required by this method; only the input features are needed.

5.4.2 Robust Bias-Aware method

The second approach we consider is the Robust Bias-Aware (RBA) classifier [20], a recently developed approach that tries to be robust to the worst-case uncertainty created by covariate shift, rather than trying to remove it. The conditional label distribution is obtained by solving a convex optimization problem. The solution has a parametric form with Lagrange multiplier θ:

$$\hat{P}_{\theta}(y|x) = \frac{e^{\frac{P_{\text{train}}(x)}{P_{\text{test}}(x)}\theta \cdot \mathbf{f}(x,y)}}{\sum_{y' \in \mathcal{Y}} e^{\frac{P_{\text{train}}(x)}{P_{\text{test}}(x)}\theta \cdot \mathbf{f}(x,y')}}.$$

The density ratio, $P_{\text{train}}(x)/P_{\text{test}}(x)$, adjusts the model to be more certain when a test data point is similar to the training data and less certain when it is not as similar. We employ a previously developed method, RuLSIF [35], to estimate the density ratio for both covariate shift approaches.

6. EXPERIMENTS

6.1 Data sets

To the best of our knowledge, we are the first to address this table-related context retrieval problem and know of no existing annotated data sets suitable for this task. We limited our study in this paper on Wikipedia pages and collected data from it. We randomly picked 36 pages that contain horizontal HTML tables with headers from Wikipedia dump[7]. We also selected 27 pages from Wiki_Manual data set which was used in previous studies [19]. Nine of the original 36 pages in Wiki_Manual were dropped because they are not suitable for our problem, e.g. the "Ian Fleming" page contains no table, the "List of newspapers in New York" page contains HTML lists rather than tables. In total there are 63 HTML web pages in our data set.

If one web page contains more than one table, we keep the largest table, which we assume contains more information. We note that in practice, since a web page may contain more than one table, instead of retrieving the relevant sentences for each table, we may want to determine for each sentence which table(s) is/are relevant. Our system can be easily adapted to this setting, because all models used in TRCR can produce probabilistic predictions. We can feed all tables in a web page to TRCR one by one. TRCR will return the probability of how likely each table-sentence pair is relevant. Then sentences can be ranked according to the probability of the relevance of each table-sentence pair, hence the most related table(s) can be found from the final ranked list. The data is annotated by two experts independently, both of whom annotate all of the instances. Each table-sentence pair is labeled as RELEVANT or IRRELEVANT, according to the definition of relevance described in Section 4. The inter-annotator agreement κ was calculated by Cohen's kappa coefficient, which adjusts for chance, ranging from [0,1], where 1 indicates perfect agreement, and 0 indicates agreement by chance. The total κ for the data set is 0.8998. In interpreting κ, Landis and Koch [17] suggest that values above 0.61 indicate substantial strength of agreement, and therefore, we believe our annotation result is consistent enough to support our conclusions. A consensus on the final labeling is made by all the annotators.

We split our data into two data sets, one for feature engineering and training, and the other strictly for testing purposes. The statistics of our data sets are shown in Table 1. Notice that our data sets are highly imbalanced, i.e., only a small portion of pairs are labeled as RELEVANT.

Data set	Training	Test
# table-sentence pairs	2633	1962
# RELEVANT pairs	415	163
# IRRELEVANT pairs	2218	1799

Table 1: Statistics of data sets.

6.2 Baselines

As one baseline, we blindly consider all of the sentences in the same section with the web table as RELEVANT. We call this baseline as the Surround Text (ST) method. We also consider applying our definitions of relevance naïvely as another baseline: Naïve Definition (ND). The algorithm of ND is listed in Algorithm 2, where input S is the sentence, T is the table, and p is the web page.

Algorithm 2 Naïve Definition
INPUT: S, T, p
OUTPUT: S and T are RELEVANT or IRRELEVANT

```
1: boolean b1 ← "table" ∈ S || "list" ∈ S;
2: boolean b2 ← any_header_token(T) ∈ S
3:                || any_cell_token(T) ∈ S;
4: boolean b3 ← any_section_heading_token(T) ∈ S;
5: boolean b4 ← any_title_token(p) ∈ S;
6: boolean b5 ← section_number(T) = section_number(S);
7:
8: if b1 || b2 || ((b3 || b4) && b5) then
9:     return RELEVANT;
10: else
11:     return IRRELEVANT;
```

Since both ST and DT are not machine learning algorithms, we check their performances on both training and test data sets, and list them in Table 2. Also because we have highly imbalanced data (i.e. the number of IRRELEVANT pairs are far larger than the number of RELEVANT pairs), we care more about the performance of RELEVANT predictions in practice. Therefore, we do not report the performance on predicting the IRRELEVANT pairs.

Baseline	Data set	F1	Precision	Recall
ST	**Training**	0.245	0.486	0.164
	Test	0.186	0.581	0.110
ND	**Training**	0.336	0.202	1.000
	Test	0.227	0.128	1.000

Table 2: Baseline performances.

From the results, we can see that although nearly half of the surrounding sentences are relevant to the table, the recall

[7] `http://dumps.wikimedia.org/enwiki/`, April 4, 2013

is very low. This is because the number of sentences (both RELEVANT and IRRELEVANT) in same section with the table is very small. In the training data set, only 140 sentences out of 2633 were in the same section as the table (68 of them are actually RELEVANT). In the test data set, only 31 sentence out of 1962 were in the same section as the table (18 of them are RELEVANT). So blindly using surrounding text would ignore a large amount of relevant information and bring nearly the same amount of noise as relevant information. This also serves as evidence of sample selection bias between training and test data sets, even though no bias was intentionally created.

We note that the recall of ND on both data sets are 1, which means to be relevant, the table and sentence pair must have some common tokens. On the other hand, low precision means that just using the token matching based on the definition of relevance without any sophisticated understanding technique is far from sufficient.

6.3 Experiments and evaluations

We evaluate the performance of Decision Trees (DT), k-Nearest Neighbors (k-NN, where k=10), Naïve Bayes (NB), SVM (linear kernel), Logistic Regression with ℓ_1 regularization (LR), Sample Reweighted (SR) logistic regression, and the Robust Bias-Aware (RBA) classifier on our data sets. Each model is trained from training data, and tuned using 10-fold cross-validation. Then we evaluate the learned models on a withheld test data set. Again, we skip the reports of performance of IRRELEVANT pairs. The performance of each algorithm on test data set is listed in Table 3.

Algorithm	F1	Precision	Recall
DT	0.489	0.447	0.540
k-NN	0.448	0.644	0.344
NB	0.484	0.369	0.706
SVM	0.685	0.696	0.675
LR	0.703	0.658	**0.755**
SR	0.674	0.617	0.742
RBA	**0.735**	**0.722**	0.748

Table 3: ML algorithm performances on test data set.

As we can see from the results in Table 3, all the Machine Learning algorithms perform better than baselines. The Decision Tree does not work particularly well with respect to both precision and recall. The k-Nearest Neighbor model has a precision of 0.644 but very low recall. On the contrary, the Naïve Bayes model has a recall of 0.706 while its precision is the lowest among all the models we learned. Using Logistic Regression, we can reach F1=0.703, which is the best model that doesn't consider sample selection bias. SVM has better precision than LR, but a lower recall, resulting in a lower F1. The consideration of sample selection bias in training data doesn't gain Sample Reweighted logistic regression any advantage over normal logistic regression model. The RBA classifier provides the second highest recall, and achieves higher precision than LR, achieving the best F1 of 0.735.

To find out the most useful features for RELEVANT prediction, we list them in descending order according to their weights of RELEVANT prediction in the learned RBA model (due to lack of space, we don't show the list here). 35 features have positive weights, which means with them an instance has a higher likelihood to be RELEVANT. However, note that using only the top features is not good enough for reaching the best performance. We experiment with models that using only top 10, 20, 30 and 35 top features, the F1's are 0.418, 0.518, 0.650, and 0.675, respectively. Comparing with F1=0.735 when using all the features, we can see that the negative-weighted features are also useful in the RELEVANT prediction.

In the feature list, the top one is the "<header> <be>" pattern that has a much more heavier weight than the others, since it directly captures the "header description" relatedness. Most of the rest top features measure tokens overlapping, either plain sentence or semantics (arguments or relation of the triple extracted by ReVerb). For instance, among the top 10 features, 8 of them indicate commonalities, either explicitly or implicitly. An explicit commonality is that there are common tokens overlapping between the sentence and the table (which is also indicated by the baseline ND, i.e. recall=1, that to be relevant there should be a certain degree of commonalities). An implicit commonality is that the sentence contains some tokens in the title of the page or the heading of the table section. As we mentioned in Section 5.3, the former captures the subject of the web page, the later captures the subject of the table, and such subject-relevances implicitly indicate the relatedness between the sentence and the table. The boolean feature that checks whether any number in the sentence matches the total number of data rows is ranked 10th, which suggests such pattern is common in summarizing the information within the web table. The feature that checks whether the sentence and the table are in the same section has a negative weight and is ranked 57th. Since the negative-weighted feature contributes IRRELEVANT prediction, it implies that a sentence can still have a large chance of being IRRELEVANT to the table even they are in the same section. It provides a support to the authors of [12]'s claim that the surrounding text often contains irrelevant information.

Commonality is important in our model, however, most of the errors are also due to it. For example, mis-classifying the IRRELEVANT sentence *"It is sometimes reported that Erdős himself has an Erdős-Bacon number of three."* as RELEVANT is mainly because it contains tokens *"Erdős"*, *"Bacon"* and *"3"* which appear in the table (see Figure 7) that actually doesn't contain the instance about the person *"Erdős"*. Solving such problem requires a much more deeper understanding of both the sentence and the table content, which we will explore in the future.

6.4 Advantages of RBA model

As we see in Table 3, RBA performs best among all the learning algorithms. In order to prove that the increase in performance from taking training data bias into account is significant, we conduct a series of experiments to compare RBA with LR. Each provides the best F1 of the models assuming sample selection bias, and no sample selection bias respectively. So as to make each pair of training and test data sets different from others while still keeping enough data points in each data set, we randomly split our training data ($TRAIN$) into 10 folds ($TRAIN_1, TRAIN_2, ...,$ $TRAIN_{10}$). Then we conduct 10 experiments, each time using 9 folds of the training data ($TRAIN - TRAIN_i$, where i = 1, 2, ..., 10) to train both LR and RBA models, and then test each model on the test data. We find that in the 10 experiments, the recall measures are almost the same on

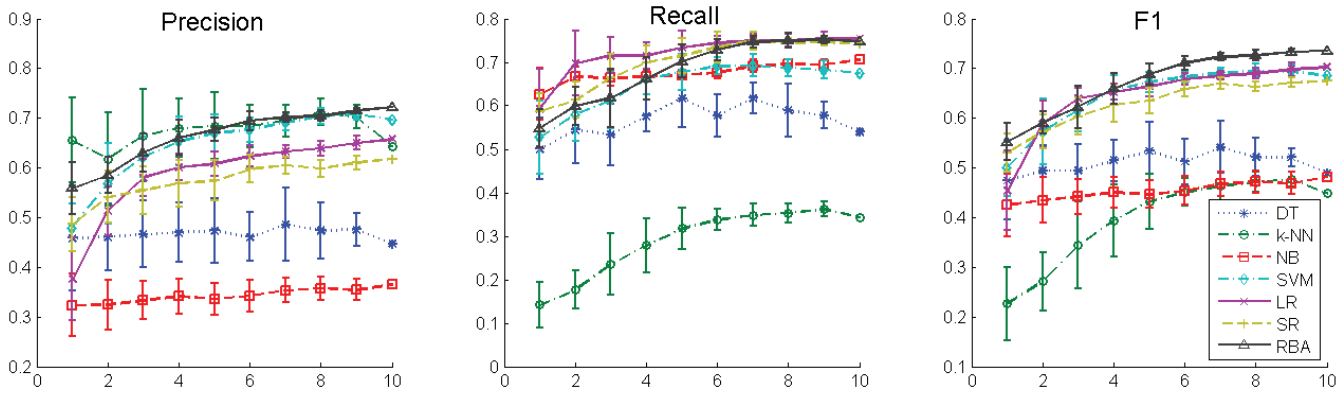

Figure 9: Performances of algorithms on size-accumulated experiments (the x-axis is the number of folds of training data).

both models, but RBA is always better on precision than the LR model, and hence achieves the better F1 scores. Both of the performance improvements of precision and F1 from RBA over LR are statistically significant, with p-values at 10^{-9} and 10^{-7} respectively.

RBA also requires less training data to reach a good performance. In order to examine the influence of training data size on the performance of each algorithm, we conduct another series of experiments. We randomly shuffle the training data 10 times. The experiments conducted each time form an experiment group $i \in [1, 10]$. We split the shuffled training data set from group i into 10 equal size folds $(TRAIN_{i1}, TRAIN_{i2}, ..., TRAIN_{i10})$. For each experiment $j \in [1, 10]$, $\bigcup_{k=1}^{j} TRAIN_{ik}$ folds are used as training data (i.e. starting with one fold of data, during each experiment, we accumulatively add one more fold into the training data). Then each learned model is tested on the whole test data set. We show the measures of performance of each algorithm in Figure 9. Those measures are averaged among the 10 experiment groups, the vertical bars at each data point indicates the corresponding standard deviation (note that under our experiment setting, we always have the whole training data set (i.e. $\bigcup_{k=1}^{10} TRAIN_{ik}$) in the last experiment in each group, hence the last standard deviations are zero). From the results, we can observe that for the same size of training data, RBA has consistent performance, and outperforms all other algorithms (its averaged F1 score is significantly better than any other algorithm's with the p-value of at most 10^{-3}). Furthermore, we can find that RBA reaches the second best F1 score (from LR) at x-axis point around 5 in the Figure. This indicates that using just half of the amount of training data, RBA can reach a better F1 score than any other experimented algorithm that uses the whole training data. In other words, in practice, with the help of RBA, we need to only label half the size of the original labeled training data to reach the same level of performance. Hence, the labeling burden for the model learning can be reduced remarkably.

7. CONCLUSION AND FUTURE WORK

In this paper, we systematically study the problem of table-related context retrieval: given a web table and the sentences within the same web page, determine for each sentence whether it is relevant to the table, where six types of relatedness are exploited to define the notion of relevance. We propose a Table-Related Context Retrieval system (TRCR) that retrieves table-related sentences. We provide the detailed description of the features we engineer for this system. Using a set of different machine learning algorithms, including a recently developed algorithm (RBA) that is robust to biases in the training data, we show that our system achieves the best F1=0.735, when RBA is used to retrieve table-related context. We also demonstrate that using RBA, a decent performance can be achieved with a smaller size of labeled training data than using traditional supervised algorithms.

There are many open avenues for further study. First, we intend to build another system that utilizes TRCR's retrieved relevant sentences of a web table as the input. The system will extract and link relations from both the table and the related sentences, and eventually discover more information and augment a knowledge base. Second, the study could be furthered by distinguishing different types of relevance. In this way, we will locate the exact part of the table that is relevant to the sentence. Third, like POS tagging and named entity recognition in the NLP research area, the structure information among sentences can be utilized, together with probabilistic graphical models (e.g. conditional random fields [16]), to improve the performance of TRCR.

8. REFERENCES

[1] A. Arnold, R. Nallapati, and W. W. Cohen. A comparative study of methods for transductive transfer learning. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 77–82. IEEE, 2007.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[3] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction for the web. In *IJCAI*, volume 7, pages 2670–2676, 2007.

[4] M. J. Cafarella, A. Halevy, and N. Khoussainova. Data integration for the relational web. *Proceedings of the VLDB Endowment*, 2(1):1090–1101, 2009.

[5] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Uncovering the relational web.

[6] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on

the web. *Proceedings of the VLDB Endowment*, 1(1):538–549, 2008.

[7] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3, 2010.

[8] C. Cortes, Y. Mansour, and M. Mohri. Learning bounds for importance weighting. In *Advances in neural information processing systems*, pages 442–450, 2010.

[9] X. L. Dong, K. Murphy, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion, 2014.

[10] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics, 2011.

[11] V. Govindaraju, C. Zhang, and C. Ré. Understanding tables in context using standard nlp toolkits. In *ACL (2)*, pages 658–664, 2013.

[12] R. Gupta, A. Halevy, X. Wang, S. E. Whang, and F. Wu. Biperpedia: An ontology for search applications. *Proceedings of the VLDB Endowment*, 7(7), 2014.

[13] E. Hatcher, O. Gospodnetic, and M. McCandless. Lucene in action, 2004.

[14] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2006.

[15] J. Jiang and C. Zhai. Instance weighting for domain adaptation in nlp. In *ACL*, volume 7, pages 264–271. Citeseer, 2007.

[16] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[17] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[18] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34. ACL, 2011.

[19] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proceedings of the VLDB Endowment*, 3(1-2):1338–1347, 2010.

[20] A. Liu and B. Ziebart. Robust classification under sample selection bias. In *Advances in Neural Information Processing Systems*, pages 37–45, 2014.

[21] F. Niu, C. Zhang, C. Ré, and J. Shavlik. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 8(3):42–73, 2012.

[22] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[23] R. Pimplikar and S. Sarawagi. Answering table queries on the web using column keywords. *Proceedings of the VLDB Endowment*, 5(10):908–919, 2012.

[24] M. F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[25] K. Raghunathan, H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. Manning. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501. ACL, 2010.

[26] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[27] A. Singhal. Introducing the knowledge graph: things, not strings. *Official Google Blog, May*, 2012.

[28] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642. Citeseer, 2013.

[29] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th WWW*, pages 697–706. ACM, 2007.

[30] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. *Proceedings of the VLDB Endowment*, 4(9):528–538, 2011.

[31] J. Wang, H. Wang, Z. Wang, and K. Q. Zhu. Understanding tables on the web. In *Conceptual Modeling*, pages 141–155. Springer, 2012.

[32] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD*, pages 481–492. ACM, 2012.

[33] R. Xia, J. Yu, F. Xu, and S. Wang. Instance-based domain adaptation in nlp via in-target-domain logistic approximation. In *Proceedings of the Twenty-Eighth AAAI*, 2014.

[34] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the 2012 ACM SIGMOD*, pages 97–108. ACM, 2012.

[35] M. Yamada, T. Suzuki, T. Kanamori, H. Hachiya, and M. Sugiyama. Relative density-ratio estimation for robust distribution comparison. In *Advances in neural information processing systems*, pages 594–602, 2011.

[36] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114. ACM, 2004.

[37] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen. Statsnowball: a statistical approach to extracting entity relationships. In *Proceedings of the 18th WWW*, pages 101–110. ACM, 2009.