Methods for Exploring and Mining Tables on Wikipedia

Chandra Sekhar Bhagavatula, Thanapon Noraset, Doug Downey
Department of Electrical Engineering & Computer Science, Northwestern University
{csbhagav, nor.thanapon}@u.northwestern.edu, ddowney@eecs.northwestern.edu

ABSTRACT

Knowledge bases extracted automatically from the Web present new opportunities for data mining and exploration. Given a large, heterogeneous set of extracted relations, new tools are needed for searching the knowledge and uncovering relationships of interest. We present *WikiTables*, a Web application that enables users to interactively explore tabular knowledge extracted from Wikipedia.

In experiments, we show that *WikiTables* substantially outperforms baselines on the novel task of automatically joining together disparate tables to uncover "interesting" relationships between table columns. We find that a "Semantic Relatedness" measure that leverages the Wikipedia link structure accounts for a majority of this improvement. Further, on the task of keyword search for tables, we show that *WikiTables* performs comparably to Google Fusion Tables despite using an order of magnitude fewer tables. Our work also includes the release of a number of public resources, including over 15 million tuples of extracted tabular data, manually annotated evaluation sets, and public APIs.

1 Introduction

Researchers have made significant strides toward automatically extracting massive, open-domain knowledge bases from Web content [1–10]. While a variety of extraction methods have been developed, the important tasks of searching, browsing, and mining the new knowledge bases have remained relatively unexplored.

This paper investigates new methods for searching and mining a knowledge base extracted from Wikipedia data tables. Consider the "electricity consumption" table found in Wikipedia (which lists total Megawatt-hours consumed for each country, and other data).[1] A user viewing this table may be curious how *other* properties of countries (e.g. CO2 emissions, GDP, etc.) are related to electricity consumption. This information need motivates the first task we consider, *relevant join*: the task of automatically identifying, given a query table $\mathbf{T}_q$, which columns from other tables would make relevant additions to $\mathbf{T}_q$. Viewed as a database join, in the *relevant join* task we first identify a column c in $\mathbf{T}_q$ to use for joining—e.g., country names in the electricity consumption table. Then, we find a distinct table $\mathbf{T}_t$ (e.g. a list of countries by GDP) with a column c' similar to c, and perform a left outer join on $c = c'$ to augment $\mathbf{T}_q$ with an automatically selected target column from $\mathbf{T}_t$ (e.g., the GDP amounts). Unlike a standard database join, the *relevant join* task requires automatically selecting the join columns and the target column to make a *relevant* addition to the query table. Automatically joining distinct tables can provide users with unified data views that aren't available in any single table on the Web.

Further, the open-domain nature of tables in Wikipedia presents an opportunity for automatically uncovering interesting or even surprising statistical relationships. The second task we investigate is the novel *correlation mining* task: determining which correlations between numeric columns are likely to be interesting or surprising, rather than trivial or spurious. For example, a user may find it interesting that electricity consumption is strongly correlated with population, and be surprised to learn it is even more strongly correlated with GDP.

In this paper, we introduce *WikiTables*, a prototype information exploration system that allows users to search, join, and mine Wikipedia tables.[2] *WikiTables* focuses on Wikipedia tables, rather than tables across the Web. Our results indicate that Wikipedia's smaller size (1.4M data tables, compared to an estimated 154 million across the Web [1]) is often outweighed by its high quality and more easily extracted semantics. In contrast to Web tables, Wikipedia's tables are explicitly distinguished from page layout, and rarely duplicated across pages.

WikiTables leverages Wikipedia's rich content and link structure as features within machine learners to search and mine the extracted tables. As one example, the semantic relatedness (SR) between Wikipedia pages estimated from the link graph [11] forms an extremely valuable feature for identifying relevant joins. On the *relevant join* task, we show that *WikiTables* more than doubles the accuracy of baselines, and that Semantic Relatedness (SR) measures account for 58% of the increase. Preliminary results on the *correlation mining* task show that our system improves F1-score over the baseline by about 26%. Finally, we also present experimental results on the previously studied *table search*, the task of

[1] `http://en.wikipedia.org/wiki/List_of_countries_by_electricity_consumption`

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

returning relevant tables in response to a given textual query, and show that *WikiTables* performs comparably to previously published results by Venetis et al. [9], *despite* using an order of magnitude fewer tables.

Our work includes the release of a publicly available prototype for each of the three tasks, along with several data resources, including over 15 million tuples of extracted tabular data, the first manually annotated test sets for the *relevant join*, *table search*, and *correlation mining* tasks, and public APIs for accessing tabular data and computing Semantic Relatedness.[3]

2 Previous Work

A wide variety of recent research is aimed at automatically extracting large bodies of structured data from the Web. Examples include systems that extract relational tuples from Web text [3–7] or from semi-structured Web sources like Wikipedia infoboxes [12, 13] or lists [14, 15]. This paper focuses on the more recent approach of extracting data from *tables* on the Web to power new Web search capabilities [1, 8–10, 16, 17].

Compared to other Web information extraction targets, table extraction offers certain advantages. Each table typically encapsulates a complete, non-redundant set of facts of a given type (e.g., an "electricity consumption" table on Wikipedia lists total Megawatt-hours consumed for each country, per-capita statistics, and the year the information was measured[4]). In extraction from free text, by contrast, systems extract facts individually, and then must deduplicate and synthesize the facts to form a complete set—a very challenging research problem (see e.g. [18]).

One closely related work to ours is found in the Google Fusion Tables system [1, 9, 10]. Our *WikiTables* system was inspired by Fusion Tables and shares similar goals of providing interactive, searchable access to tabular information. However, *WikiTables* is distinct in its focus on Wikipedia tables, and on the mining of tabular data. We introduce the novel *relevant join* task, which shares commonalities with the *schema complement* task studied in [10] but is distinct in that *relevant join* is aimed at automatically identifying specific columns of relevant data, rather than related tables as in [10]. We also present first investigation into the *correlation mining* task on extracted tables. Also, unlike the Fusion Tables work, our table corpus, APIs, and evaluation data with human relevance judgments are all publicly available.

Yakout et. al. introduced the task of Attribute Discovery in [19], which involves finding columns of attributes from tables for a given query set of entities. Our *relevant join* task differs from Attribute Discovery in an important way: in Attribute Discovery, the input is a set of entities, whereas in *relevant join* the input is a table. Thus, unlike in Yakout et. al., the columns returned for *relevant join* by *WikiTables* must be relevant not only to the set of entities, but also to the context in which the set of entities is found. For example, we would expect the most relevant columns to add to a country GDP table would differ from the most relevant columns for an Olympic Medal table, even though the entities are the same in both cases (countries).

Finally, our experiments reveal that a Semantic Relatedness (SR) measure that estimates the relatedness between Wikipedia topics using the link graph [11, 20] is especially valuable for the *relevant join* task. Experimenting with other SR measures that utilize other inputs, beyond Wikipedia links (e.g., [21, 22]), is an item of future work.

3 Task Definitions

In this section, we formally define our three tasks: *relevant join*, *correlation mining* and *table search*. All three tasks utilize a corpus of tables extracted from Wikipedia, so we begin by formally defining the table extraction process.

3.1 Table Extraction from Wikipedia

WikiTables scans all Wikipedia articles for tables. Each table found in an article is extracted and converted into an $m \times n$ matrix of cells (details in Section 4.1). We refer to these matrices as *normalized tables*. Each cell in the matrix holds data from one cell in the table, consisting of all string values and links to Wikipedia concepts contained in the table cell.

The set of all normalized tables extracted from Wikipedia forms our corpus, $\mathcal{T}$. Each normalized table $\mathbf{T}_i \in \mathcal{T}$, having m rows and n columns is represented as a list of column vectors:

$$\mathbf{T}_i = (\mathbf{c}_1^i \mathbf{c}_2^i \dots \mathbf{c}_n^i)$$
$$\text{where } \forall k \in [1, n] \;\; \mathbf{c}_k^i \text{ is a vector of length } m$$

3.2 Relevant Join

Relevant join is the task of finding *columns* that can be added to a given table and ranking them in descending order of relevance to the given table.

Consider a user viewing table $\mathbf{T}_q$, the *QueryTable*, as shown in Figure 1. An informative column to add to this table could be the population data of each country. This data exists in a different table, denoted as $\mathbf{T}_t$ (the *TargetTable*). The "Country" column of $\mathbf{T}_q$ and the "Nation" column of $\mathbf{T}_t$ contain nearly identical data, which suggests the two tables can be joined on these columns. The columns from $\mathbf{T}_q$ and $\mathbf{T}_t$ that contain similar data are called the *SourceColumn* ($\mathbf{c}_s^q$) and *MatchedColumn* ($\mathbf{c}_m^t$) respectively. All columns from $\mathbf{T}_t$ except $\mathbf{c}_m^t$ can be added to $\mathbf{T}_q$. We refer to any column that can be added to a *QueryTable* from a *TargetTable* as a *CandidateColumn*, denoted by $\mathbf{c}_c^t$. The final table, $\mathbf{T}_f$, is created by appending $\mathbf{T}_q$ with the "Population" column from $\mathbf{T}_t$. A *CandidateColumn* that is added to a *QueryTable* is referred to as an *AddedColumn*, $\mathbf{c}_a^t$.

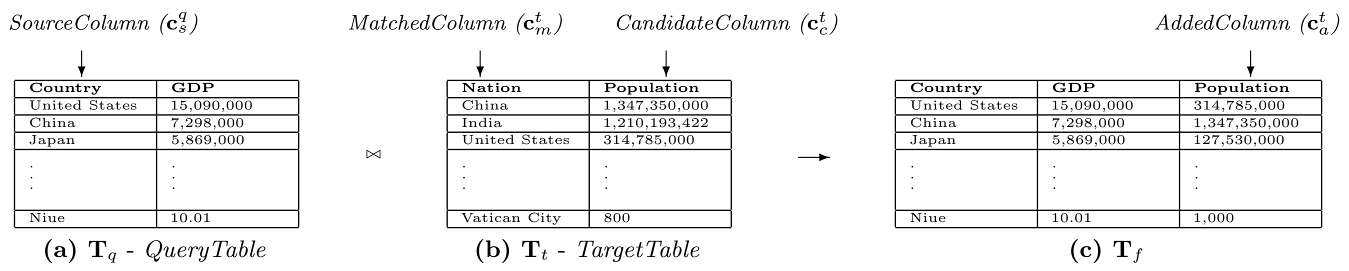
Formally, FindRelevantJoin($\mathbf{T}_q, \mathcal{T}$) takes a query table $\mathbf{T}_q$ and the corpus of tables $\mathcal{T}$ as inputs, and returns a ranked list of triplets denoting relevant joins. Each triplet consists of the *SourceColumn* ($\mathbf{c}_s^q$), the *MatchedColumn* ($\mathbf{c}_m^t$) and the *CandidateColumn* ($\mathbf{c}_c^t$). Triplets ($\mathbf{c}_s^q, \mathbf{c}_m^t, \mathbf{c}_c^t$) are ranked in decreasing order of relevance of $\mathbf{c}_c^t$ to $\mathbf{T}_q$. In each triplet, $m \neq c$, columns $\mathbf{c}_m^t, \mathbf{c}_c^t \in \mathbf{T}_t$ and $\mathbf{c}_s^q \in \mathbf{T}_q$. Columns $\mathbf{c}_s^q$ and $\mathbf{c}_m^t$ are chosen such that they contain nearly identical data in their cells. The pre-processed set of (*SourceColumn*, *MatchedColumn*) pairs across the corpus $\mathcal{T}$ is referred by $\mathcal{M}$.

Example 1: Figure 2 shows a table from the *"List of countries by GDP (nominal)"* Wikipedia page.[5] This table is the query table, $\mathbf{T}_q$. The column *"GDP (PPP) \$Billion"* is the *AddedColumn* $\mathbf{c}_a^t \in \mathbf{T}_t$, where $\mathbf{T}_t$ is the *"List of countries*

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

[4] `http://en.wikipedia.org/wiki/List_of_countries_by_electricity_consumption`

[5] `http://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)#List`



SourceColumn ($\mathbf{c}_s^q$)
MatchedColumn ($\mathbf{c}_m^t$)
CandidateColumn ($\mathbf{c}_c^t$)
AddedColumn ($\mathbf{c}_a^t$)
$\bowtie$
$\rightarrow$

Country	GDP
United States	15,090,000
China	7,298,000
Japan	5,869,000
⋮	⋮
Niue	10.01

(a) $\mathbf{T}_q$ - *QueryTable*

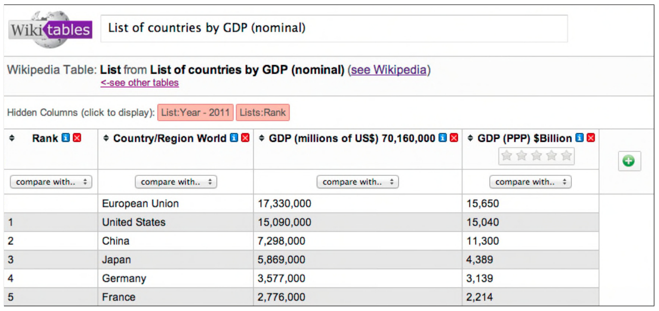
Nation	Population
China	1,347,350,000
India	1,210,193,422
United States	314,785,000
⋮	⋮
Vatican City	800

(b) $\mathbf{T}_t$ - *TargetTable*

Country	GDP	Population
United States	15,090,000	314,785,000
China	7,298,000	1,347,350,000
Japan	5,869,000	127,530,000
⋮	⋮	⋮
Niue	10.01	1,000

(c) $\mathbf{T}_f$

Figure 1: Joining a *QueryTable* $\mathbf{T}_q$, (that contains GDP information) with a *TargetTable* $\mathbf{T}_t$, (that contains Population data) to get $\mathbf{T}_f$. Country column from $\mathbf{T}_q$ is the *SourceColumn* and the Nation column from $\mathbf{T}_t$ is the *MatchedColumn*.



List of countries by GDP (nominal)

Wikipedia Table: **List** from **List of countries by GDP (nominal)** (see Wikipedia)
<-see other tables

Hidden Columns (click to display): List:Year - 2011 Lists:Rank

Rank	Country/Region World	GDP (millions of US$) 70,160,000	GDP (PPP) $Billion
	European Union	17,330,000	15,650
1	United States	15,090,000	15,040
2	China	7,298,000	11,300
3	Japan	5,869,000	4,389
4	Germany	3,577,000	3,139
5	France	2,776,000	2,214

Figure 2: *WikiTables* column addition in action: *WikiTables* shows the table containing list of countries and their GDP (nominal) stats. The first 3 columns belong to this table. *WikiTables* adds the right-most column, "GDP (PPP) Billion", to this table.

by GDP (PPP)" table.[6] The data in the "Country/Region" column in $\mathbf{T}_q$ matches the data from the "Country" column in $\mathbf{T}_t$. These two columns from $\mathbf{T}_q$ and $\mathbf{T}_t$ are the match columns $\mathbf{c}_s^q$ and $\mathbf{c}_m^t$ respectively.

The *relevant join* task involves two primary challenges. First, we wish to select an *AddedColumn* that lists attribute values for the entities in the *SourceColumn*. In practice this can be challenging. Consider a hypothetical table containing columns, "City," "Country," and "Mayor." Here, "Mayor" is a property of "City," and not of "Country." Thus, if we erroneously select "Country" as a *MatchedColumn* and "Mayor" as an *AddedColumn*, this will lead to a poor join. Secondly, we wish to choose *AddedColumns* that are relevant to the query table. For example, the "prisoners" column is relevant to the incarceration table in Figure 2, whereas other attributes of countries (Olympic gold medal winners, for example) are far less relevant. The *relevant join* task requires identifying this distinction automatically, for a given table. Both of the above challenges are addressed by our trained models for the *relevant join*, which are described in the following section.

3.3 Correlation Mining

Correlation Mining is the novel task of determining which correlations between numeric table columns are"interesting."

Formally, FINDCORRELATIONS($\mathcal{M}$) takes the set $\mathcal{M}$, consisting of pairs of matching columns, and generates a set $\mathcal{P}$, of triplets ($\mathbf{c}_s, \mathbf{c}_m, type$). $\mathbf{c}_s$ and $\mathbf{c}_m$ are numeric columns. Column $\mathbf{c}_s$ belongs to the same table as *SourceColumn* and $\mathbf{c}_m$ belongs to the same table as *MatchedColumn*, where (*SourceColumn*, *MatchedColumn*) is a pair in $\mathcal{M}$.

The third element, *type*, is one of the following three categories: (i) *Unrelated:* Signifying that there could not be a reliable relation between the numeric columns, irrespective of the correlation coefficient, or that the correlation is insignificant, e.g. the correlation of GDP with the years in which a country hosted the World Taekwondo Championships (ii) *Plausible:* If a correlation is intuitive – for example, GDP correlates highly with the imports of a country. (iii) *Surprising:* If a user finds the correlation surprising – for example, the GDP of a country correlates slightly positively with its linguistic diversity. We note that interestingness is inherently subjective, and rating a correlation as *Surprising* is largely dependent on the annotator's background. More extensive user studies that measure how the categorization of attributes may differ across different user groups is an item of future work.

3.4 Table Search

Table search is the task of returning a ranked list of tables for a given textual query. Formally, TABLESEARCH($q, \mathcal{T}$) takes as input a query, q, and the corpus of tables, $\mathcal{T}$ and returns a list of tables in decreasing order of relevance to q. For example, the top three results returned by *WikiTables* for **TableSearch**(`country population`, $\mathcal{T}$) [7] are *1*) List of countries by population *2*) List of sovereign states and dependent territories by population density *3*) World population.

4 WikiTables - System Description

In this section, we describe our *WikiTables* system, and its *relevant join*, *correlation mining* and *table search* capabilities. *WikiTables* performs these tasks using a corpus of tables extracted in advance from Wikipedia. Parsing Wikipedia tables accurately requires some care. We first describe our extraction method and evaluate its accuracy. We then describe how *WikiTables* utilizes the extracted tables for the *relevant join*, *correlation mining* and *table search* tasks.

4.1 Extracting Wikipedia Tables

Wikipedia offers free download of all articles. The data-source for *WikiTables* is the English Wikipedia XML dump available online.[8] This XML dump contains data in Wiki Markup which encases tables between the '{|' and '|}' tags.[9] However, some Wikipedia pages may also contain tables that are *not*

[6] `http://en.wikipedia.org/wiki/List_of_countries_by_GDP_(PPP)#Lists`

[7] `http://downey-n1.cs.northwestern.edu/tableSearch/`

[8] `http://dumps.wikimedia.org/enwiki/`

[9] `http://en.wikipedia.org/wiki/Help:Wiki_markup`

found within these table tags, but get rendered to web users at runtime by expanding *templates*.[10] We found nearly 643,000 Wikipedia pages that contained about 1.4 million tables that were either enclosed within the tags or rendered by expanding templates.

We restrict extraction to tables that belong to the HTML class `wikitable` since it is the only class used to denote data tables on Wikipedia.[11] Filtering on this class attribute allows us to ignore other parts of a page that are rendered as tables – such as infoboxes, table of contents or meta-data boxes – which we do not wish to include in our table corpus.

The resulting set of matrices form our corpus of extracted normalized tables, $\mathcal{T}$.

To evaluate the quality of our table extraction, we picked 50 random pages (using Wikipedia's random page generator[12]) that contained at least 1 table. The total number of tables found on these 50 pages was 111. *WikiTables* extracted 82 out of these 111 tables ($recall = 0.74$). The total number of cells in the 82 tables is 6404, of which *WikiTables* extracted 6383 cells accurately ($precision = 99.7\%$).[13] This is much more precise than table extractors that crawl the full Web, which are faced with lower-quality content and the challenging problem of interpreting when `<table>` tags indicate relational data rather than page layout. In fact, the previous WebTables system achieved a precision of only 41% in extracting relational data tables [23], whereas all tables extracted by *WikiTables* contain relational data.

Hence, we find that *WikiTables* extracts tables at very high precision at the cost of some recall. Most of the loss in recall is due to data tables missing the class attribute `wikitable`. Identifying and fixing such errors is part of future work. Our experiments in Section 5 show that this level of extraction precision and recall is sufficient to build a system that is comparable to other implementations of *table search*.

4.2 Relevant Join Implementation

In this section we describe how *WikiTables* performs the *relevant join* task. *WikiTables*'s implementation of the *relevant join* task requires identifying *CandidateColumns* and computing Semantic Relatedness (SR) for the candidates in real-time. We begin by describing pre-processing which enables efficient execution at query time.

4.2.1 Pre-Processing

Our table corpus $\mathcal{T}$ contains more than 7.5 million columns in all the tables in $\mathcal{T}$ which makes finding joins at runtime prohibitively inefficient. In order to find relevant joins in realtime, we pre-process tables in $\mathcal{T}$ to identify which tables can be joined with a given table. This pre-processing allows *WikiTables* to quickly identify *CandidateColumns* for a given query table efficiently, and then use machine learning models (described in the following section) to rank these *CandidateColumns*.

We pre-compute pairs of matched columns, $(\mathbf{c}_s^i, \mathbf{c}_m^j)$ for columns $\mathbf{c}_s^i \in \mathbf{T}_i$ and $\mathbf{c}_m^j \in \mathbf{T}_j$. Here $\mathbf{c}_s^i$ is the *SourceColumn* and $\mathbf{c}_m^j$ is the *MatchedColumn*. The percentage of values in $\mathbf{c}_s^i$ found in $\mathbf{c}_m^j$ is called the *MatchPercentage*. It must be noted that *MatchPercentage* is not a commutative property of a pair of columns, $(\mathbf{c}_s^i, \mathbf{c}_m^j)$.

To reduce the number of columns that are considered as candidates for joins, we use heuristics to select only those columns that (i) are non-numeric (ii) have more than 4 rows of data (iii) have an average string length greater than 4. This prevents joining tables on columns that contain serial numbers, ranks, etc. In our experience, using these heuristics improves the quality of joins dramatically.

After filtering out columns with the heuristics, we are left with about 1.75 million columns. We compute these matches exhaustively, to obtain the *MatchPercentage* for each pair of columns. Matches that have a *MatchPercentage* greater than 50% are added to our corpus of matches $\mathcal{M}$.

We have nearly 340 million matches in $\mathcal{M}$, the set of matching pairs. Section 4.2.3 explains how $\mathcal{T}$ and $\mathcal{M}$ are used at runtime to find candidate columns to add to a table and rank them by relevance.

4.2.2 Semantic Relatedness in ***WikiTables***

A *Semantic Relatedness* (SR) measure takes as input two concepts, and returns a numeric relatedness score for the concept pair. Our experiments demonstrate that preferring joins between page concepts with higher SR results in significant performance improvements on the *relevant join* task.

In *WikiTables*, we use the *MilneWitten* Semantic Relatedness measure [11], which estimates the relatedness of Wikipedia concepts (each Wikipedia page is treated as a "concept"). *MilneWitten* is based on the intuition that more similar Wikipedia pages should share a larger fraction of inlinks. We use the *MilneWitten* implementation from Hecht et al. [20], an enhancement to the original measure that emphasizes prominent links within the Wikipedia page "gloss" and learns parameters of the measure based on hand-labeled relatedness judgments.

Utilizing SR at runtime within *WikiTables* requires quickly computing the relatedness between the page containing the query table and all other pages containing candidate columns. Naively computing in-link intersections with every candidate column at query time would be intractable. To solve this performance issue, we pre-compute all non-zero relatedness scores for all concepts on Wikipedia, and store these in memory. The data store is compressed (by quantizing SR scores to 256 bins and using simple variable-byte encoding), and occupies about 30GB of memory. We have provided access to this data store through a public API.[14]

4.2.3 Finding Relevant Joins

In *WikiTables*, the user begins by selecting a table to view. This table is the input to the FindRelevantJoins algorithm illustrated in Algorithm 1. FindRelevantJoins takes a query table $\mathbf{T}_q$ and the table corpus $\mathcal{T}$ as inputs, and returns a set of triplets. A triplet contains a *SourceColumn*, a *MatchedColumn* and a *CandidateColumn*, denoted $(\mathbf{c}_s^q, \mathbf{c}_m^t, \mathbf{c}_c^t)$. These triplets are ranked on the basis of the estimated relevance of adding $\mathbf{c}_c^t$ to $\mathbf{T}_q$. From the top ranked triplet, $\mathbf{c}_c^t$ can be added to $\mathbf{T}_q$ through a left outer join between $\mathbf{T}_q$ and $\mathbf{T}_t$ ON $\mathbf{c}_s^q = \mathbf{c}_m^t$.

These triplets are formed by first getting all matches from $\mathcal{M}$, in decreasing order of *MatchPercentage* such that one of the columns from $\mathbf{T}_q$ is the *SourceColumn* (the function GetMatchPairs in Algorithm 1). All columns from the

[10] `http://en.wikipedia.org/wiki/Help:Template`
[11] `http://en.wikipedia.org/wiki/Help:Wikitable`
[12] `http://en.wikipedia.org/wiki/Special:Random`
[13] `http://downey-n1.cs.northwestern.edu/public/`
[14] `http://downey-n2.cs.northwestern.edu:8080/wikisr/sr/sID/69058/langID/1`

Algorithm 1 Pseudo-code to find relevant joins for a given table

```
function FindRelevantJoins($\mathbf{T}_q$, $\mathcal{T}$)
    $C_{Matches} \leftarrow \emptyset$        ▷ Set of (SourceColumn , MatchedColumn) pairs
    for all $\mathbf{c} \in$ GetColumns($\mathbf{T}_q$) do        ▷ GetColumns($\mathbf{T}_q$) returns all columns in $\mathbf{T}_q$
        $C_{Matches} \leftarrow C_{Matches} \cup$ GetMatchPairs($c$)        ▷ Get Pairs from $\mathcal{M}$ in descending order of match percentage, for a source column
    end for
    Sort$_{MatchPercent}(C_{M_c})$        ▷ Sort (SourceColumn , MatchedColumn) pairs in decreasing order of MatchPercent
    $C_{candidates} \leftarrow \{\}$        ▷ Set of (SourceColumn , MatchedColumn, CandidateColumn) triplets
    for all $(\mathbf{c}_s^i, \mathbf{c}_m^j) \in C_{Matches}$ do
        for all $x \in$ GetColumns(TableOf($\mathbf{c}_m^j$)) $\wedge\ x \neq c_{\mathrm{m}}$ do        ▷ TableOf($\mathbf{c}_m^t$) returns table, $\mathbf{T}_t$ to which $\mathbf{c}_m^t$ belongs
            $C_{candidates} \leftarrow C_{candidates} \cup (\mathbf{c}_s^i, \mathbf{c}_m^j, x)$
        end for
    end for
    $C_F \leftarrow$ ClassifyJoins($C_{candidates}$)        ▷ Classify triplets as relevant or non-relevant using a trained classifier
    $C_R \leftarrow$ RankJoins($C_F$)        ▷ Rank triplets in $C_F$ using a ranking model
    return $C_R$
end function
```

matched column's table except the matching column are considered as candidate columns.

Candidate columns are first classified as either *relevant* or *non-relevant* using a trained classifier model, and then ranked using a ranking model. The classifier discards low quality column additions. We used the Weka implementation [24] of the Logistic Regression model as our classifier, and a linear feature based Coordinate Ascent ranking model, implemented by RankLib.[15] Both models use features listed in Table 1. The models were trained on small set of hand labeled examples as described in Section 5.1.1.

Entities in the *SourceColumn* may be present more than once in the *MatchedColumn*. In this case, all values from the *AddedColumn* that correspond to a single entity in the *MatchedColumn* are grouped together and added to the query table, i.e. we display all values that we join to within a single cell. A feature ("avgNumKeyMap" in Table 1) measures how many values each cell in the added column contains, on average. Our results (Section 5.1) show that one-to-many joins are not preferred by users—there is a strong negative correlation between the avgNumKeyMap feature and the relevance rating of a column.

4.3 Correlation Mining Implementation

WikiTables uses the set of column matches $\mathcal{M}$ to find correlations between numeric columns. For each pair $(\mathbf{c}_s^i, \mathbf{c}_m^j) \in \mathcal{M}$, *WikiTables* finds all pairs of numeric columns $(\mathbf{n}^i, \mathbf{n}^j)$, such that $\mathbf{n}^i \in$ TableOf($\mathbf{c}_s^i$) and $\mathbf{n}^j \in$ TableOf($\mathbf{c}_s^j$), and calculates Pearson correlation coefficient between the numeric columns. Table 2 lists the features used to build machine learning models to classify each pair of correlated columns into one of three types described in Section 3.3.

4.4 Table Search Implementation

The dataset used for *table search* is the set of normalized tables, $\mathcal{T}$, extracted from Wikipedia. Our goal is to find the most relevant tables from $\mathcal{T}$ for a given textual query q, using contents of the table and the text around the table. To incorporate textual features of the page containing the table, we use an existing search service.[16] At runtime, a user's query is first sent to the search service, and result URLs restricted to the *en.wikipedia.org* domain are retrieved. Using $\mathcal{T}$, we obtain tables found in each of the top 30 Wikipedia pages returned for a query. A trained linear ranking model (learned

Name	*Description*
matchPercentage	*MatchPercentage* between *SourceColumn* and *MatchedColumn*
avgNumKeyMap	Average number of times a key in the *SourceColumn* is found in the *MatchedColumn*
srRelate	SR measure between the page containing the *SourceColumn* and the page containing the *MatchedColumn*
srcAvgLen	Average length of data in the *SourceColumn*
srcDistinctValFrac	Fraction of distinct values in the *SourceColumn*
srcNumericValFrac	Fraction of columns in *SourceTable* that are numeric
tgtAvgLen	Average length of data in the *CandidateColumn*
tgtIsNum	Boolean, 1 if *CandidateColumn* is numeric
tgtDistinctValFrac	Fraction of distinct values in the *CandidateColumn*
inLink	Number of inlinks to the page to which *TargetTable* belongs
outLink	Number of outlinks from the page to which *TargetTable* belongs
srcTableColIdx	Boolean feature: 1 if *SourceColumn* is among the first three columns of the table
matchedTableColIdx	Boolean feature: 1 if *TargetColumn* is among the first three columns of the table

Table 1: Features used by *WikiTables* for the *relevant join* task

Name	*Description*
corel	Correlation Coefficient between numeric columns
sr	Semantic Relatedness measure between the pages containing the matched pair of columns
se	Standard error of the correlations coefficient
numSamples	Number of samples considered in the calculation of correlation coefficient
zScore	zScore of correlations for a given source column

Table 2: Features used by *WikiTables* for the *correlation mining* task

with Coordinate Ascent [25]) ranks these tables using four types of features described in Table 3. We train our ranker on a small set of hand-labeled examples, as described in Section 5.3.1.

5 Experiments

In this section, we present our evaluation of *WikiTables* on the three tasks: *relevant join*, *correlation mining* and *table search*.

[15] `http://people.cs.umass.edu/~vdang/ranklib.html`
[16] `http://developer.yahoo.com/boss/search/`

Name	***Description***
Page Features	
yRank	Rank in Yahoo!'s results for query
inLinks	Number of in-links to page
outLinks	Number of out-links from
pageViews	Number of page views
Table Features	
numRows	Number of rows
numCols	Number of columns
emptyCellRatio	Fraction of empty cells
Table+Page Features	
sectionNumber	Wikipedia Section index the table occurs in
tableImportance	Inverse of number of tables on page
tablePageFraction	Ratio of table size to page size
Query Features	
qInPgTitle	Ratio: Number of query tokens found in page title **to** total number of tokens
qInTableTitle	Ratio: Number of query tokens found in table title **to** total number of tokens

Table 3: Features used by *WikiTables* for the *table search* task

We describe the datasets, comparison methods, and metrics used for this evaluation, along with results obtained. We also present an analysis of the results obtained for each task. Experiments show that *WikiTables* beats baseline methods on the *relevant join* and *correlation mining*. *WikiTables* also performs comparably to previous work on the *table search* task, despite using an order of magnitude fewer tables. We also demonstrate that Semantic Relatedness measures defined over Wikipedia concepts are especially valuable for the *relevant join* task.

5.1 Relevant Join

5.1.1 Datasets

We evaluated *WikiTables* on a dataset that consists of tables to which columns can be added. To provide a non-trivial number of columns to rank, we performed a weighted random selection of 20 Wikipedia articles that contained at least one table, where weight of a page is the number of times a column from that page occurs in our match corpus $\mathcal{M}$ as a *SourceColumn*. The random list of articles used for evaluation are listed here.[17]

The WikiTables UI allows users to rate automatically joined columns on a scale of 1 to 5. To train the machine learning models, for *relevant join*, described in Section 4.2, 4 users posted 593 ratings on 82 distinct tables, mutually exclusive from the test set. For the classifier, all ratings of 3 and above were considered *relevant* (positive class of the classifier). Input for the ranker is the actual rating given by the user. For evaluation, two annotators rated the top 4 columns added to each table from pages in the test set, on a scale of 1 to 5, indicating relevance of an added column to the query table. Inter annotator agreement, measured by Spearman's rank correlation coefficient between ratings of the two annotators for each query, was found to be 0.73.

5.1.2 Comparison Methods

To our knowledge, *WikiTables* is the first system that finds and ranks relevant columns that can be added to a given table. Previous work has looked into using *Schema Complement* as an indicator of relatedness of tables and using this relatedness to improve table search [10]. GOOGLE FUSION TABLES allows adding columns to a table from other tables, but a user selects the *subject* column manually and also selects the column(s) to be added. Thus, to evaluate *WikiTables* on the *relevant join* task, we created a baseline method (BASE) and compared it with our implementation, *WikiTables*. BASE simply ranks *CandidateColumns* in decreasing order of *MatchPercentage*. In an ablation study to evaluate the impact of *Semantic Relatedness* on this task, we created a system (*WikiTables−SR*) which ignores the *srRelate* feature from both the classifier and ranking models.

5.1.3 Metrics

To evaluate *WikiTables* on this task, we used the standard information retrieval metrics of Discounted Cumulative Gain (*DCG*) [26] and Normalized Discounted Cumulative Gain (*nDCG*). Both of these metrics are designed to give greater importance to occurrences of more useful results higher in a ranked list. *DCG* at a particular rank position p is given by:

$$\mathbf{DCG_p} = \sum_{i=1}^{p} (2^{rel_i} - 1)/(\log_2(i+1)) \tag{1}$$

where rel_i is the rating of the i^{th} ranked result table. *nDCG* is defined as the ratio between *DCG* of the ranked list retrieved and *DCG* of the ideal ranking possible. This is given by:

$$\mathbf{nDCG_p} = (DCG_p)/(IDCG_p) \tag{2}$$

nDCG metric is calculated for the top 4 columns added by a method to each query table. We choose to evaluate on the top 4 as this is a typical number of added columns that fits in a single browser window. The normalizing factor in this case is the ideal ranking of the union of columns added by the three methods that are being compared. Performance of each method can be estimated by taking the average of *nDCG* values across all queries.

We also measure the accuracy of columns that are added to a table by a method. We define this metric as:

$$\mathbf{Acc_m} = \frac{\#\text{ columns with rating} \geq 3}{\#\text{ columns added by method m}} \tag{3}$$

5.1.4 Results and Analysis

The results of our evaluation are summarized in Table 4, which lists the average accuracy of the four columns added by each method, and the average *DCG* and *nDCG′* values of results retrieved for all query tables. Both BASE and *WikiTables−SR* added columns to 106 tables and *WikiTables* added columns to 103 tables. Results show that compared to BASE, *WikiTables* more than doubles the fraction of relevant columns retrieved. Further, Semantic Relatedness measures defined over Wikipedia concepts account for approximately 58% of this improvement (over *WikiTables−SR*).

Model	**cols added**	**acc.**	**DCG@4**	**nDCG′@4**
BASE	423	0.29	11.38	0.43
WikiTables−SR	414	0.43	13.71	0.48
WikiTables	410	**0.62**	**30.71**	**0.76**

Table 4: Results of comparison of three methods for the *relevant join* task. *WikiTables* more than doubles the accuracy of added columns as compared to the baseline Base. Using Semantic Relatedness feature results in a significantly improved performance in *WikiTables* as compared to *WikiTables−SR*

Table 5 lists the correlation between features and user rating for the *relevant join* task. Results show that columns

[17]`http://downey-n1.cs.northwestern.edu/public/randomQSet.html`

containing numeric data make more relevant additions to a table than other non-numeric ones. A greater value of Semantic Relatedness, more distinct values in the source column, and a higher match percentage between matched columns leads to higher-quality of added columns.

Feature	**Correlation with rating**
tgtIsNum	0.285
srRelate	0.253
avgNumKeyMap	-0.231
srcDistinctValFrac	0.209
srcTableColIdx	-0.202
matchPerc	0.167
outLink	-0.161
inLink	-0.157
srcNumericValFrac	0.134

Table 5: Spearman's rank correlation coefficient of *relevant join* features with column rating. Only features with significant correlations ($p < 0.01$) are listed. Correlation is calculated over 485 pairs.

The results described above show that the *Semantic Relatedness* measure between Wikipedia pages is a very valuable indicator of relatedness of which column should be added to a given table. For example, for a table of the top 10 largest power producing facilities, *WikiTables−SR* adds an unrelated column containing the number of Olympic medals won by countries in swimming. On the other hand, *WikiTables*, which uses the Semantic Relatedness features, adds highly relevant information about the annual CO_2 emissions of countries.

5.2 Correlation Mining

5.2.1 Datasets

We evaluated *WikiTables* on the *correlation mining* task using 100 randomly sampled pairs of numeric columns from the set of correlations $\mathcal{P}$. These 100 pairs were manually classified into one of three categories: *Unrelated*, *Plausible*, or *Surprising* correlations.

5.2.2 Comparison Methods

We tried different models and compared their performance on our dataset. The models used are: ZeroR for baseline, Logistic Regression, SVM Classifier, 1-Nearest Neighbor and 3-Nearest Neighbor classifiers. Each classifier is used to predict one of the three classes: *Unrelated*, *Plausible* or *Surprising*.

5.2.3 Metrics

We performed 10-fold cross validation over these 100 pairs of labeled columns. We choose the standard F1 score as our metric for evaluation.

5.2.4 Results and Analysis

Table 6 shows the accuracy and F1 score of each model. We find that 1-vs-All Logistic Regression model has the best performance.

Using our best model, i.e. Logistic Regression, we performed an ablation study by removing each of the five features. Results are tabulated in Table 7.

5.3 Table Search

We compare *WikiTables* with previously published work by Venetis et al. [9] (referred to further as Table). We define metrics that are used to measure relative performance of table search methods and present results that show that

Model	**Accuracy**	**F1**
ZeroR (Baseline)	66%	0.525
SVM Classifier	65%	0.52
1-Nearest Neighbor	64%	0.62
3-Nearest Neighbor	71%	0.657
Logistic Regression	**71%**	**0.661**

Table 6: Preliminary Results on the Correlation Mining task. All classifiers classify data into 4 classes (using 1-vs-All classification where applicable)

Feature removed	**Accuracy**	**F1**
corel	70%	0.649
sr	68%	0.626
se	70%	0.639
numSamples	71%	0.661
zScore	72%	0.66

Table 7: Ablation Study

WikiTables outperforms Table in the *table search* task. The results of the *table search* experiments also show that even with a much smaller dataset of tables, *WikiTables* satisfies more user queries aimed at retrieving tables from the Web.

5.3.1 Datasets

As we compare *WikiTables* with Web-based extractors, it is important that we do not bias the evaluation in favor of information from Wikipedia. To prevent this bias, all of our *table search* evaluation query workloads are drawn from previous work on searching tables from the Web at large.

Our first test set D1 consists of a set of 100 queries of the form (C, P), where C is a string that denotes a *class* of instances, and P is also a string that denotes a *property* associated with these instances. This dataset is taken from previous work by Venetis et al. [9]. For each query in the test set of 100 (C, P) pairs, Venetis et al. rated the top 5 tables retrieved by a table search method as either *right on* (if it had all information about a large number of instances of the class and values for the property), *relevant* (if it had information about only some of the instances, or of properties that were closely related to the queried property) or *irrelevant*. Some systems of table search (e.g. Goog , GoogR - described in the following section), used by [9] to compare with Table, return documents and not just tables. Thus, annotators also annotated if a result was found *in a table*. On the other hand, *WikiTables* only returns tables as results of a search query.

We replicated this evaluation method on *WikiTables*, hand-labeling results for each of the 100 queries. Because *WikiTables* takes a single string as its input rather than a (C, P) pair, we simply concatenate the class and property strings to form the textual query for our system. More sophisticated handling of classes and properties within *WikiTables* is an item of future work.

We measured inter-annotator agreement on a held out set of 8 queries (from Sarma et al. [10]). Two raters annotated each table returned for queries in this set on a scale of 0 to 5. The average Spearman's correlation coefficient between ratings of the annotators across all queries was 0.71, which we believe is sufficiently high for such a subjective task.

For training the *WikiTables*'s table search ranker (Section 4.4), we utilized a training set of about 350 labeled examples on 5 queries, disjoint from the test set queries.

Method	All Ratings				Ratings by Queries				Metrics		
	Total	(a)	(b)	(c)	Some Result	(a)	(b)	(c)	$\mathbf{P_{tq}}$	$\mathbf{R_{tq}}$	$\mathbf{F1_{tq}}$
WikiTables	500	54	88	142	100	**35**	**63**	**63**	0.63	**0.63**	**0.63**
Table	175	69	98	93	49	24	41	40	**0.82**	0.4	0.54
Document	399	24	58	47	93	13	36	32	0.34	0.32	0.33
Goog	493	63	116	52	100	32	52	35	0.35	0.35	0.35
GoogR	156	43	67	59	65	17	32	29	0.45	0.29	0.35

Table 8: Comparing results of Table and *WikiTables*. *WikiTables* outperforms Table with a much higher recall and $F1$ score. Results of Document, Goog and GoogR have been re-printed from [9]. The columns under All Ratings present the number of results that were rated to be (a) *right on*, (b) *right on* or *relevant*, and (c) *right on* or *relevant* and *in a table*. The Ratings by Queries columns aggregate ratings by queries: the sub-columns indicate the number of queries for which a table in the top 5 results got a rating with (a) *right on*, (b) *right on* or *relevant*, and (c) *right on* or *relevant* and *in a table*.

5.3.2 Comparison Methods

We compare *WikiTables* with previous work by Venetis et al.. Our primary evaluation compares performance on data set `D1` with previously published results from [9]. The best-performing method from that work, Table, automatically annotates all tables in the corpus with labels, which indicate what a table is *about*, and ranks these labels on importance. The table search system within Table takes queries of the form (C, P), where C is a class name and P is a property, and returns a ranked list of tables. To create this ranked list, Table considers tables in the corpus that have a class label C in the top-k class labels in its annotations. Table then ranks these tables based on a weighted sum of a number of signals, some of which are derived from the property P from the query. The weights were determined using a training set of examples. In their published results, Venetis et al. compared performance of Table with three other methods: 1. Goog : the results returned by www.google.com, 2. GoogR: the intersection of the table corpus (from Table) with the top-1,000 results returned by GOOG, and 3. Document: document-based approach proposed in [1]. We include results of the performance of all methods compared in [9], for completeness.

5.3.3 Metrics

The primary metric we use to compare *WikiTables* with previously published results on data set `D1` is simply the number of queries for which a given method returned a result in its top 5 that was *relevant* or *right on* and *in a table*. Venetis et al. also evaluated *precision* on the table search task [9]. We consider precision to be a secondary metric for this task (it is rare in information retrieval tasks to distinguish between returning poor results and returning no results), but following [9] we also evaluate on precision and F1 metrics:[18]

$$\mathbf{P_{tq}} = \frac{\#\text{ queries with } \textit{right on} \text{ or } \textit{relevant} \text{ tables}}{\#\text{ queries for which results returned}} \quad (4)$$

$$\mathbf{R_{tq}} = \frac{\#\text{ queries with } \textit{right on} \text{ or } \textit{relevant} \text{ tables}}{\#\text{ queries}} \quad (5)$$

$$\mathbf{F1_{tq}} = (2 * \mathbf{P_{tq}} * \mathbf{R_{tq}})/(\mathbf{P_{tq}} + \mathbf{R_{tq}}) \quad (6)$$

5.3.4 Results and Analysis

Table 8 shows the results of our experiment comparing *WikiTables* with Table using the Dataset `D1`. While Table only produced a result for 49 of the 100 queries, *WikiTables* retrieved tables for all 100 queries. Furthermore, Table returned a *right on* or *relevant* table in the top 5 results for only 40 queries, whereas *WikiTables* returned a relevant table for 63 queries—a 58% increase.

Our results show that on query workloads from previous work on Web table search, the higher quality of Wikipedia data combined with our machine learning approach results in *WikiTables* outperforming Web table search systems. The text surrounding a table is more relevant for table search when tables are drawn from high-quality, topically-focused pages like those on Wikipedia.

Feature	Correlation with rating
tablePageFraction	0.41
numRows	0.40
tableCaption ContainsQuery	0.31
sectionNumber	-0.19
inlinks	-0.16
numCols	0.156

Table 9: Spearman's rank correlation coefficient between *table search* features and user ratings on tables retrieved for text queries. Only features with Significant correlations ($p < 0.01$) are listed.

Table 9 lists the Spearman's rank correlation coefficients between features and user ratings of tables. The features in bold represent statistically significant correlations. From the table, it can be observed that tables that cover larger fraction of a page tend to be better and that bigger tables are better. One surprising observation is that the number of inlinks has a significant negative correlation with result quality. This is because prominent Wikipedia pages (with more inlinks) do not necessarily contain better tables. For example, the "List of counties in Maryland" page, which has only 928 inlinks, contains much higher-quality tabular data than the "Maryland" page that has about 29,000 inlinks. The "List of prolific inventors" page, which has 67 inlinks, contains much better tabular data than the "Alexander Graham Bell" page that has about 2600 inlinks.

In our evaluation of *WikiTables* on Dataset `D1`, there were 37 queries for which *WikiTables* did not retrieve any *right on* or *relevant* tables in the top 5 results. We analyzed the types of queries for which *WikiTables* failed to retrieve relevant tables. The query classes for which *WikiTables* did not retrieve accurate tables are the ones that are open-ended, i.e. the number of entities that belong to these classes is very large. Examples include *movie stars*, *guitars*, *football clubs*, *beers*, *clothes*, etc. Since these classes contain large numbers of entities, they are generally not found in just one single table. They rather tend to be sub-categorized into smaller

[18]Our metrics are similar to but differ slightly from those of [9]; the precise metrics they use depend on which methods retrieved accurate tables for which queries, which is not known to us.

groups. Such *open-ended* queries accounted for 17 out of the 37 queries in which *WikiTables* did not retrieve any *right on* or *relevant* tables.

6 Conclusions and Future Work

In this paper, we introduced *WikiTables*, a system that extracts, searches, and automatically joins Wikipedia tables. We showed that leveraging high-quality textual and tabular content on Wikipedia is valuable for the *relevant join*, *correlation mining*, and *table search* tasks. On the novel *relevant join* and *correlation mining* tasks, our machine learning methods are found to significantly outperform baselines. Results on *table search* show that the smaller number of tables on Wikipedia is outweighed by their higher quality and more easily extractable semantics.

In future work, we plan to investigate ways to combine *WikiTables* with the broader-coverage Google Fusion Tables to improve the precision and recall of each. We also wish to investigate richer interactive data mining over the table corpus, that goes beyond the two-column correlations studied in this paper. Wikipedia tables and Web tables include an unprecedentedly broad collection of entity properties, presenting a novel opportunity to generate insights from data and make predictions. For example, one could ask whether the myriad of properties of countries found across Wikipedia tables might aid in predicting how a country's unemployment rate will change over time.

7 Acknowledgments

This work was supported in part by NSF Grant IIS-1016754 and DARPA contract D11AP00268.

8 References

[1] 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*, 2008.

[2] Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics, 2007.

[3] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.M. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of the 13th WWW*, 2004.

[4] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Organizing and searching the world wide web of facts-step one: the one-million fact extraction challenge. In *Proceedings of the National Conference on Artificial Intelligence*, 2006.

[5] M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. *Open information extraction for the web*. PhD thesis, University of Washington, 2009.

[6] R. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *Annual meeting-association for Computational Linguistics*, 2007.

[7] Y. Fang and K.C.C. Chang. Searching patterns for relation extraction over the web: rediscovering the pattern-relation duality. In *Proceedings of the fourth WSDM*, 2011.

[8] S. Chakrabarti, S. Sarawagi, and S. Sudarshan. Enhancing search with structure. *IEEE Data Eng. Bull*, 2010.

[9] 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*, 2011.

[10] A. Das Sarma, L. Fang, N. Gupta, A. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu. Finding related tables. In *Proceedings of the 2012 SIGMOD*. ACM, 2012.

[11] I.H. Witten and D. Milne. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*, 2008.

[12] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia-a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2009.

[13] F. Wu and D.S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM CIKM*, 2007.

[14] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. *Proceedings of the VLDB Endowment*, 2009.

[15] R.C. Wang and W.W. Cohen. Iterative set expansion of named entities using the web. In *Data Mining, 2008. ICDM'08. Eighth ICDM*, 2008.

[16] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. Towards domain-independent information extraction from web tables. In *Proceedings of the 16th international conference on World Wide Web*, 2007.

[17] B. Chan, L. Wu, J. Talbot, M. Cammarano, and P. Hanrahan. Vispedia: Interactive visual exploration of wikipedia data via search-based integration. *IEEE TVCG*, 2008.

[18] A. Yates and O. Etzioni. Unsupervised resolution of objects and relations on the web. In *Proceedings of NAACL HLT*, 2007.

[19] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the 2012 international conference on Management of Data*, pages 97–108. ACM, 2012.

[20] B. Hecht, S.H. Carton, M. Quaderi, J. Schöning, M. Raubal, D. Gergle, and D. Downey. Explanatory semantic relatedness and explicit spatialization for exploratory search. In *Proceedings of the 35th international ACM SIGIR*, 2012.

[21] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th IJCAI*, 2007.

[22] J. Hoffart, S. Seufert, D.B. Nguyen, M. Theobald, and G. Weikum. Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM CIKM*, 2012.

[23] M.J. Cafarella, A.Y. Halevy, Y. Zhang, D.Z. Wang, and E. Wu. Uncovering the relational web. *WebDB*, 2008.

[24] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 2009.

[25] D. Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 2007.

[26] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM (TOIS)*, 2002.