Entity Discovery and Annotation in Tables

Gianluca Quercini, Chantal Reynaud-Delaître

Entity Discovery and Annotation in Tables

Gianluca Quercini
Université Paris-Sud XI
Laboratoire de Recherche en Informatique (LRI)
gianluca.quercini@lri.fr

Chantal Reynaud
Université Paris-Sud XI
Laboratoire de Recherche en Informatique (LRI)
chantal.reynaud@lri.fr

ABSTRACT

The Web is rich of tables (e.g., HTML tables, spreadsheets, Google Fusion Tables) that host a considerable wealth of high-quality relational data. Unlike unstructured texts, tables usually favour the automatic extraction of data because of their regular structure and properties. The data extraction is usually complemented by the *annotation* of the table, which determines its semantics by identifying a *type* for each column, the *relations* between columns, if any, and the *entities* that occur in each cell.

In this paper, we focus on the problem of discovering and annotating entities in tables. More specifically, we describe an algorithm that identifies the rows of a table that contain information on entities of specific *types* (e.g., *restaurant*, *museum*, *theatre*) derived from an ontology and determines the cells in which the names of those entities occur. We implemented this algorithm while developing a faceted browser over a repository of RDF data on points of interest of cities that we extracted from Google Fusion Tables.

We claim that our algorithm complements the existing approaches, which annotate entities in a table based on a pre-compiled reference catalogue that lists the types of a finite set of entities; as a result, they are unable to discover and annotate entities that do not belong to the reference catalogue. Instead, we train our algorithm to look for information on previously unseen entities on the Web so as to annotate them with the correct type.

Categories and Subject Descriptors

H.2 [**Database management**]: Miscellaneous

General Terms

Algorithms

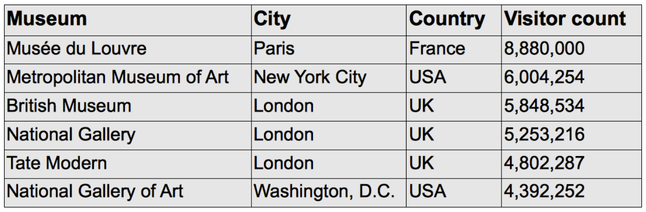
Keywords

Entity discovery in tables, Entity annotation in tables, Google Fusion Tables

1. INTRODUCTION

The Web is rich of high-quality tabular data [3], including, just to mention a few, HTML tables occurring in Web pages, spreadsheets, and tables hosted by *Google Fusion Tables* (GFT), a data management service recently launched by Google [9]. Besides being rich of data, tables also show some interesting properties that, unlike unstructured texts, favour the *automatic extraction* of information, in particular (Figure 1):

- Each cell contains some *atomic information* (e.g., the name of a museum or a city) that is readily available for extraction; unstructured texts need to be segmented and analyzed to find atomic information.
- The cells in a single column have an *homogeneous content*; for example, the first column in Figure 1 only contains names of museums.
- A table provides a *summary* of the most important information (e.g., name, city) of an entity (e.g., a museum), which in a text are usually buried in a sea of words and therefore need to be identified.
- Tables often contain data that are unlikely to be found in a text. If we were to report statistics, such as the number of visitors per year in museums, we would rather use a table instead of filling a text with numbers.



Museum	City	Country	Visitor count
Musée du Louvre	Paris	France	8,880,000
Metropolitan Museum of Art	New York City	USA	6,004,254
British Museum	London	UK	5,848,534
National Gallery	London	UK	5,253,216
Tate Modern	London	UK	4,802,287
National Gallery of Art	Washington, D.C.	USA	4,392,252

Figure 1: Tables feature properties that favour the extraction of information.

Being able to automatically extract data from a table is rather useless if their semantics is unknown; this is why the extraction of data is usually complemented by the *annotation* of the table, which usually goes through three main steps (not necessarily in the following order) [14, 22]:

a. Determine the *type(s)* of each column. This step consists of understanding, for example, that the second column in Figure 1 contains names of cities.

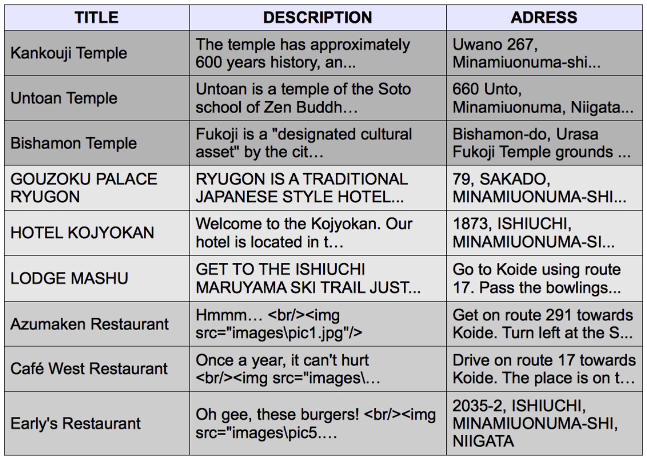
b. Find any *relationship* between the columns. For example, the first and second column in Figure 1 are linked by the relation *locatedIn*.

c. Identify the *entities* that occur in the cells.

In this paper, we are interested in the problem of discovering and annotating entities in tables, which relates to step c., while taking a different stance from existing approaches. We observed that state-of-art algorithms annotate entities in a table based on pre-compiled catalogues that contain a finite collection of entities as well as information as to their types. These catalogues are usually obtained from existing knowledge bases [14, 15, 23], ontologies [12] or information automatically extracted from Web pages [22]. As a result, these algorithms can only annotate entities that are listed in the catalogue (the *known* entities), but are unable to discover new (or *unknown*) entities. One might argue that:

1. If a table contains a mix of known and unknown entities, the type of the latter can be inferred from that of the former, based on the fact that the content of the cells in a single column is homogeneous. For instance, if the only known entities in the first column in Figure 1 are "Musée du Louvre", "Metropolitan Museum of Art" and "National Gallery of Art", which are all museums, chances are that also the remaining entities are museums too.

2. Information across different catalogues can be merged in order to obtain a larger catalogue.



TITLE	DESCRIPTION	ADRESS
Kankouji Temple	The temple has approximately 600 years history, an...	Uwano 267, Minamiuonuma-shi...
Untoan Temple	Untoan is a temple of the Soto school of Zen Buddh...	660 Unto, Minamiuonuma, Niigata...
Bishamon Temple	Fukoji is a "designated cultural asset" by the cit...	Bishamon-do, Urasa Fukoji Temple grounds ...
GOUZOKU PALACE RYUGON	RYUGON IS A TRADITIONAL JAPANESE STYLE HOTEL...	79, SAKADO, MINAMIUONUMA-SHI...
HOTEL KOJYOKAN	Welcome to the Kojyokan. Our hotel is located in t...	1873, ISHIUCHI, MINAMIUONUMA-SI...
LODGE MASHU	GET TO THE ISHIUCHI MARUYAMA SKI TRAIL JUST...	Go to Koide using route 17. Pass the bowlings...
Azumaken Restaurant	Hmmm... <br/><img src="images\pic1.jpg"/>	Get on route 291 towards Koide. Turn left at the S...
Café West Restaurant	Once a year, it can't hurt <br/><img src="images\...	Drive on route 17 towards Koide. The place is on t...
Early's Restaurant	Oh gee, these burgers! <br/><img src="images\pic5....	2035-2, ISHIUCHI, MINAMIUONUMA-SHI, NIIGATA

Figure 2: Tables containing information on entities of three different types.

As for the first argument, the entities that occur in the cells of the same column do not necessarily have the same fine-grained type; consider the first column in Figure 2, which contains names of temples (rows 2-4), hotels (rows 5-7) and restaurants (rows 8-10). If the only known entities in the first column were the temples, one would erroneously conclude that all the entities in the column are temples as well.

The second argument entails a lot of challenges, first and foremost the entity reconciliation and schema matching between different catalogues, which needs to be correctly addressed for the resulting catalogue to be error-free. Moreover, it is not clear how many catalogues need to be merged in order to have an acceptable coverage of entities for an annotation task. In our experiments, we verified that only 22% of the entities in our dataset of tables are actually represented in either Yago [19], DBpedia [1] or Freebase [1], which are the richest open datasets available to date. Recently, a new knowledge base has been proposed, called *Probase*, which contains 2.7 million concepts and is estimated to include concepts/entities that occur in more than 80% of the Web searches [24]. However, we could not verify the number of the entities in our tables that are also represented in *Probase* because its current release is only available for internal use [2].

Based on these considerations, we describe an algorithm that, given a table:

1. Identifies the *rows* that contain information on entities of a specific *type* (e.g., *restaurant*, *museum*, *theatre*, *rock band*, *singer...*) obtained from an ontology, and

2. Determines the *cells* that contain the *names* of those entities.

For example, given the table in Figure 2 and the type *restaurant*, our algorithm should determine that rows 8-10 contain information on restaurants whose names are listed in the first column. To this extent, we train our algorithm to gather information on unknown entities from the Web so as to annotate them with the correct type. More specifically, we enable our algorithm to query a Web search engine with the content of a cell and use the obtained results to determine whether the cell contains the name of an entity of a certain type. Each result consists of a link to a Web page, its title and a brief summary of its content, usually referred to as *snippet*; our algorithm uses a text classifier to determine whether a snippet describes an entity of a certain type. In a sense, when confronted with unknown entities, the algorithm mimics the way humans discover new knowledge by looking up information on the Web. This way, the algorithm is able to discover and annotate new entities that do not necessarily belong to a pre-compiled catalogue. We note that our algorithm is not in competition with the existing approaches at all; if anything, it complements them. Indeed, one can still use them to annotate entities that belong to an offline catalogue, without the need of accessing a web search engine which inevitably entails a latency time, and resort to our algorithm only to discover and annotate unknown entities.

Our algorithm is motivated by specific application needs that arose while studying a framework for advancing the technological state of the art in efficient Open Data management in digital cities. This framework is part of the research project *DataBridges: Data Integration for Digital Cities*, a 2011-2012 activity within the "Digital Cities of the Future" action line of the *EIT ICT Labs* KIC. More specifically, we developed in partnership with the *DFKI* - the *German Research Centre for Artifical Intelligence* - a faceted browser [3] over a RDF repository of points of interest (POIs), such as restaurants and museums, of cities around the world [17, 18]. We created the repository by automatically extracting data

[1]`http://www.freebase.com`

[2]`http://research.microsoft.com/en-us/projects/probase/release.aspx`

[3]`http://digitaleveredelung.lolodata.org:8080/DigitalCities/page/index.html`

from GFT tables; the discovery and annotation of entities in tables, which is the focus of this paper, is an important part of the extraction process.

GFT is a popular Web application provided by *Google* to allow people, including those with no database expertise, to manage their data [9]. Being contributed by millions of people over the Internet, GFT is a large fast-growing collection of tables with data from a number of different domains and may become a widely-used platform to publish tabular data in the years to come; the Australian government already publishes its datasets on GFT [4]. Although our algorithm is general enough to handle any type of table that can be found in the Web, in this paper we also show how we used some specific features of GFT to solve our annotation problem in the context of our application. We chose GFT as our source of data for different reasons:

- GFT tables feature a rich collection of information across different types of POIs, as opposed to other data sources, such as Wikipedia, where some types (e.g. *restaurants* and *hotels*) are under-represented;
- GFT maintains an index which favours the retrieval of tables that contain information on specific types of POIs;
- GFT tables have a neat, regular structure, compared to other tabular data, such as spreadsheets and Web tables, which facilitates the annotation and extraction of information; for instance, no column can branch into multiple subcolumns, like in spreadsheets.
- GFT provides an API that allows applications to query tables by using SQL.

In summary, we claim the following key contributions:

- We describe an algorithm that complements existing approaches by discovering and annotating new entities in tables that do not necessarily belong to a pre-compiled catalogue of entities.
- We show that the type of an entity can be determined by querying a Web search engine with the name of the entity and using a text classifier over the snippets returned by the search engine.
- We explore the properties of Google Fusion Tables, which has the potential to become a very rich repository of tabular data, to improve the annotation of entities in tables.

The remainder of the paper is organized as follows. After a short outline of related work in Section 2, we describe in greater detail the main features of GFT in Section 3. In Section 4 we better formalize the problem that we study in this paper and in Section 5 we describe our algorithm. Finally, the results of the experiments that we conducted to evaluate our algorithm are discussed in Section 6. Section 7 concludes the presentation.

2. RELATED WORK

The approaches to (fine-grained) entity annotation of which we are aware fall broadly into two categories — *context-dependent* and *context-independent*.

The former assume that entities occur in textual documents and are surrounded by a local context that may give useful insights as to the type of the entities themselves [4, 6, 7]. Fleischman et al. [6] and Ganti et al. [7] acknowledge that the local context in documents is sometimes insufficient and introduce the notion of a *multi-context*, which is created by enriching the local context with semantic information derived from external resources (respectively, *WordNet* and a corpus of documents).

Since contextual information in tables are often scarce, context-independent approaches, that make no assumption as to the context of an entity, are a better fit for annotating entities in tables. Ni et al. [16] harness the richness of the Linked Open Data (LOD) to create a type knowledge base that they use to compute the probability that an entity has a given type. However, while the LOD is a rich collection of RDF triples that express facts on millions of entities, many types of entities are still under-represented, as witnessed by the "Linked Data Shopping List", documented on the website *linkeddata.org* [5]. The approach proposed by Giuliano [8] is independent of a catalogue and therefore can discover new entities, but assumes that the entities to classify are already available (e.g. in a list); in order to annotate entities in a table, we first need to find the rows and the cells that contain those entities.

Entity annotation in tables is part of a broader task known as *table annotation*, which calls for the identification of a correct type for each column and the relationships between columns [11, 12, 14, 15, 20, 22, 23]. Automatic solutions to the problem have been proposed which, however, assign annotations to tables based on a pre-compiled catalogue of entities, types or relationships [14, 22, 23], LOD datasets [15], or ontologies [12]. Evidently, none of these approaches address the problem of annotating entities that do not exist in the reference catalogue that they use.

Finally, a closely related work to ours is described by Guo et al. [10], who propose a tool termed *ITEM* for enriching RDF knowledge bases with new entities extracted from Google Fusion Tables. This is achieved by adding to a RDF knowledge base the tuples of tables whose schema matches the one of the knowledge base. The key difference from our work is that we do not assume to have a RDF knowledge base that guides the identification of new entities in GFT tables.

3. GOOGLE FUSION TABLES

As defined by Gonzalez et al. [9], *Google Fusion Tables* (GFT for short) is a "cloud-based data management and integration service" launched by Google in June 2009 with the primary goal of making database technology accessible to a broad class of users, including those with limited expertise [25]. The rapid growth of GFT fulfilled the hopes of its creators that the ease of use of GFT would motivate people to publish their table data. Although we did not find any estimate on the actual number of publicly available tables in GFT, we have noticed a steady increase over time in the number of results returned in response to search queries.

Being a crowdsourcing system, as pointed out by Doan et al. [5], and actively updated and maintained by Google, GFT has got what it takes to become one of the largest repositories of tables on the Web. From the semantic web

[4]`http://data.gov.au/tag/google-fusion-tables/`

[5]`http://linkeddata.org/linked-data-shopping-list`

point of view, GFT may therefore soon turn out to be an extremely valuable source of high-quality data, that can be retrieved, annotated and extracted. Moreover, GFT has other advantages over other table data available online, such as spreadsheets and Web tables. Perhaps the greatest advantage is that columns are assigned a type (TEXT, NUMBER, LOCATION and DATE), which greatly helps the identification of the semantics of the content of the table.

4. PROBLEM DEFINITION

Let T be a table with n rows and m columns where each cell is denoted as $T(i, j)$, $1 \leq i \leq n$ and $1 \leq j \leq m$ being the index of the row (respectively, the column) of the cell. We assume that T can be modelled as a bi-dimensional array with $n \times m$ cells; this assumption limits our scope to tables where no column branches into subcolumns. We also assume that the content of the table is in English, leaving the interesting problem of multilingualism in tables to future work.

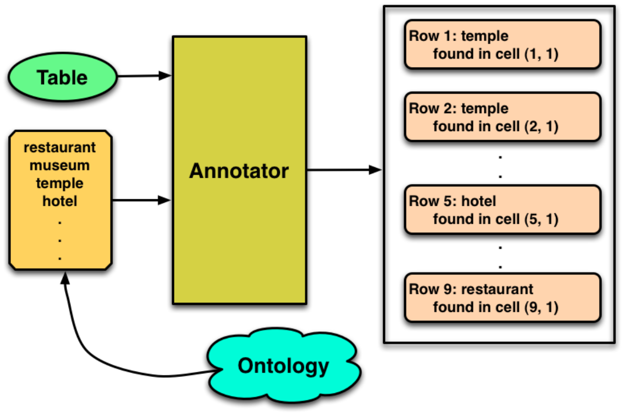
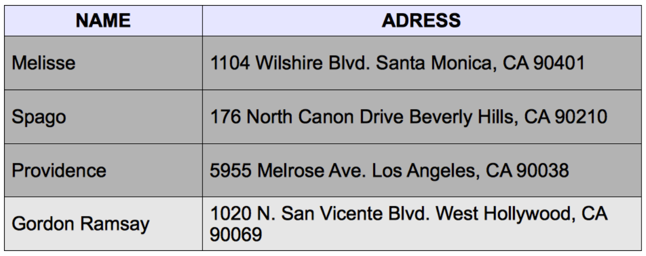


Table
restaurant
museum
temple
hotel
Annotator
Ontology
Row 1: temple found in cell (1, 1)
Row 2: temple found in cell (2, 1)
Row 5: hotel found in cell (5, 1)
Row 9: restaurant found in cell (9, 1)

Figure 3: Problem definition. The output refers to the table of Figure 2.

Figure 3 shows a schema of the problem that we intend to solve. Given a table T and a set of *types*, that correspond to *concepts* of an ontology (e.g. restaurants, museums, theatres...), our annotator must determine the rows that contain information on entities of the given types and the cells that contain the names of those entities.

Tables are often cryptic, in the sense that they do not provide strong contextual evidence as to the semantics of their content; as a result, the annotation of entities in tables often turns out to be a hard task for humans too. For



NAME	ADRESS
Melisse	1104 Wilshire Blvd. Santa Monica, CA 90401
Spago	176 North Canon Drive Beverly Hills, CA 90210
Providence	5955 Melrose Ave. Los Angeles, CA 90038
Gordon Ramsay	1020 N. San Vicente Blvd. West Hollywood, CA 90069

Figure 4: Excerpt of a table with limited context.

instance, the table shown in Figure 4 does not provide any clue to indicate that its first column contains references to restaurants. The headers of the columns are ambiguous as they can refer to any entity that has a name and an address. Therefore, for our annotation task we cannot rely on the presence of meaningful column headers or table captions.

Nonetheless, tables have some properties that can be harnessed for our annotation task; more specifically, we use the principle of *column homogeneity*, which is based on the observation that the cells in a single column have homogeneous data types. For instance, in Figure 4 all cells of the first column contain references to entities (in this case, restaurants) while the second column contains postal addresses.

5. OUR ALGORITHM

The rationale of our algorithm is that the scarce context of tables can be enriched with the wealth of information that are available on the Web. More specifically, our algorithm determines whether a cell contains the name of an entity of type t as follows (Figure 5):

1. It submits the content of the cell to a Web search engine;

2. It collects the top-k *search results*, each consisting of a link to a Web page, its title and a short summary of its content, often referred to as a *snippet*. Only results in English are considered.

3. It finds out whether the collected snippets are descriptions of an entity of type t by using a text classifier.

In a sense, the algorithm mimics the way humans discover new knowledge, as most of us resort to the Web to get information about entities that we do not know.

Here we are confronted with two challenges. First, we observe that querying a Web search engine is a costly operation due to the time needed to submit the query and receive the answer; as a result, it is not a good idea to submit a query for every cell of the table, especially when it is very large. Second, the snippets obtained as a result of a query may not be informative enough to determine whether a cell contains a reference to an entity of a certain type; this may lead to erroneous annotations, which need to be corrected. Based on these considerations, our algorithm consists of three major steps:

1. **Pre-processing.** The cells whose content is not likely to contain the names of entities of the given types are ruled out. This is done by looking at the syntactic properties of the content of each cell, as well as the GFT types of the columns in which they occur.

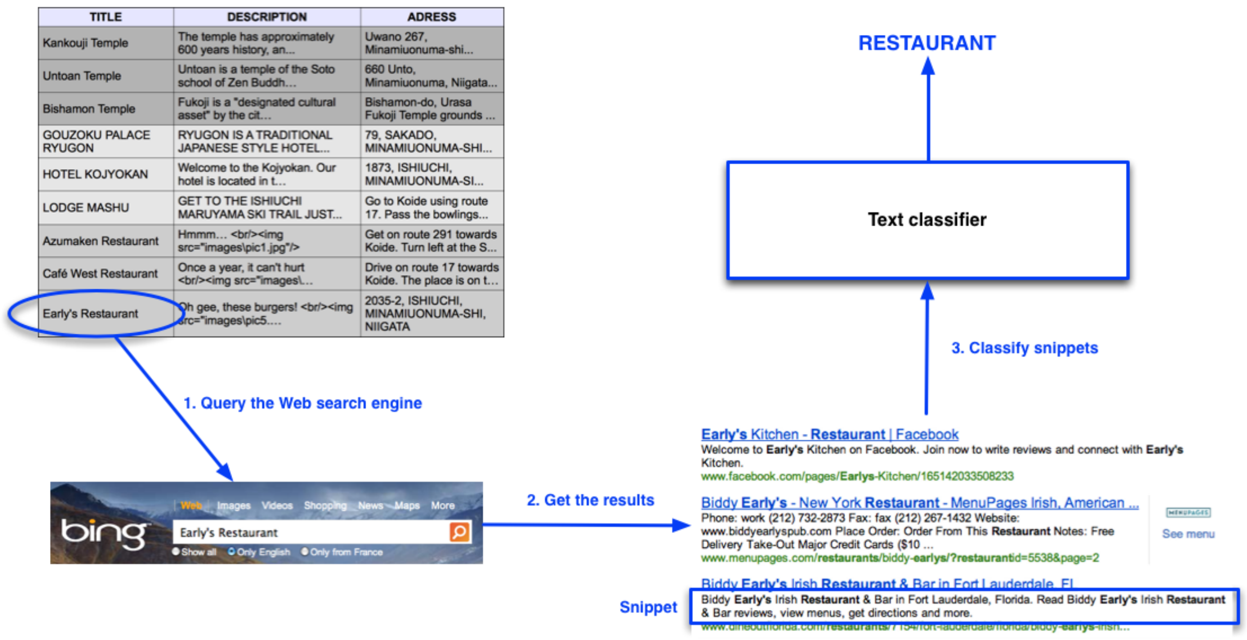
2. **Annotation.** The content of the remaining cells is submitted to a Web search engine as described above.

3. **Post-processing.** The incorrect annotations are eliminated based on the principle that the data types in the cells of a single column is homogeneous in well-formed tables.

We detail each step in the remainder of this section.

5.1 Pre-processing

The content of some cells may feature syntactic regularities that can be used to determine that they do not contain



TITLE	DESCRIPTION	ADRESS
Kankouji Temple	The temple has approximately 600 years history, an...	Uwano 267, Minamiuonuma-shi...
Untoan Temple	Untoan is a temple of the Soto school of Zen Buddh...	660 Unto, Minamiuonuma, Niigata...
Bishamon Temple	Fukoji is a "designated cultural asset" by the cit...	Bishamon-do, Urasa Fukoji Temple grounds ...
GOUZOKU PALACE RYUGON	RYUGON IS A TRADITIONAL JAPANESE STYLE HOTEL...	79, SAKADO, MINAMIUONUMA-SHI...
HOTEL KOJYOKAN	Welcome to the Kojyokan. Our hotel is located in t...	1873, ISHIUCHI, MINAMIUONUMA-SI...
LODGE MASHU	GET TO THE ISHIUCHI MARUYAMA SKI TRAIL JUST...	Go to Koide using route 17. Pass the bowlings...
Azumaken Restaurant	Hmmm... <br/><img src="images\pic1.jpg"/>	Get on route 291 towards Koide. Turn left at the S...
Café West Restaurant	Once a year, it can't hurt <br/><img src="images\...	Drive on route 17 towards Koide. The place is on t...
Early's Restaurant	Oh gee, these burgers! <br/><img src="images\pic5....	2035-2, ISHIUCHI, MINAMIUONUMA-SHI, NIIGATA

Early's Kitchen - Restaurant | Facebook
Welcome to Early's Kitchen on Facebook. Join now to write reviews and connect with Early's Kitchen.
www.facebook.com/pages/Earlys-Kitchen/165142033508233

Biddy Early's - New York Restaurant - MenuPages Irish, American ...
Phone: work (212) 732-2873 Fax: fax (212) 267-1432 Website: www.biddyearlyspub.com Place Order: Order From This Restaurant Notes: Free Delivery Take-Out Major Credit Cards ($10 ...
www.menupages.com/restaurants/biddy-earlys/?restaurantid=5538&page=2

Biddy Early's Irish Restaurant & Bar in Fort Lauderdale, Fl
Biddy Early's Irish Restaurant & Bar in Fort Lauderdale, Florida. Read Biddy Early's Irish Restaurant & Bar reviews, view menus, get directions and more.

RESTAURANT
Text classifier
3. Classify snippets
1. Query the Web search engine
bing
Web Images Videos Shopping News Maps More
Early's Restaurant
Show all Only English Only from France
2. Get the results
See menu
Snippet

Figure 5: Schema of the algorithm.

names of entities without querying the search engine. In the case of GFT tables, we can also use information as to the types of each column. More specifically, we consider the following:

- Cells containing values that follow a certain pattern, that is usually captured by regular expressions. Examples are phone numbers, URLs, email addresses, numeric values and geographic coordinates.
- Cells containing long values, such as verbose descriptions (e.g., those in the second column of Figure 2).
- Cells that belong to columns with a specific GFT type such as Location, Date, or Number.

If the algorithm wants to identify the cells that contain names of restaurants, it does not need to consider those whose content matches the regular expression for URLs or email addresses. On the other hand, if the algorithm is looking for phone numbers or URLs, it can quickly find them without resorting to a web search engine.

5.2 Annotation

For all the cells whose content does not match the regularities described in the previous subsection, our algorithm needs to use the snippets obtained from a Web search engine to find out whether they contain references to entities of a given type. As a search engine we used *Microsoft Bing*, which provides an API that imposes less restrictions on the allowance of daily queries than other major search engines. Here we are confronted with two major challenges.

First, snippets are usually very short, as most of them are less than 20 words long; how can such a short text be used to determine whether $T(i,j)$ refers to an entity of type t?

Second, the content of $T(i,j)$ may be ambiguous, which means that submitting it as a query to a search engine may return results that do not reflect its semantics. For example, referring to Figure 4, "Melisse" may refer to a restaurant, as well as to a French contemporary Jazz label; therefore, a query to the search engine is likely to retrieve snippets related to both interpretations.

As for the first issue, Giuliano (2009) proposes to compute the similarity between the snippets obtained for an entity e and those obtained for entities of type t in order to decide whether e is an entity of type t as well [8]. However, while Giuliano assumes that the entities are already identified, we do not know whether $T(i,j)$ contains an entity (e.g. a restaurant) or a phrase (e.g. a review of a restaurant) that are likely to be described by similar snippets; as a result, chances are that a review of a restaurant is classified as a reference to en entity of type restaurant. Instead, we use text classification techniques to determine whether a snippet is the description of an entity of a given type t. Note that this requires the training of a text classifier on a set of snippets that describe entities of type t, for each type that we want to annotate. In order to make our approach scalable for annotating entities of many types, we need a method to gather a training set in a way that involves as little human intervention as possible.

As for the second issue, we investigate an approach that effectively solves the ambiguity problem in the context of our application, which, as explained in Section 1, needs data on points of interest of cities around the world. Indeed, tables that have information on these entities typically contain their addresses or their geographic coordinates or, at least, a reference to the city where they are located. Therefore, the query that is submitted to the search engine can be augmented with this spatial information in order to disambiguate it. A more general solution to the ambiguity problem would be clustering the results returned by the search engine and classify separately the snippets that belong to the different clusters. We do not explore this point in this paper, which we leave for future work.

We now describe in details the solution to these two issues.

5.2.1 Using the Snippets to Annotate Entities

Given a set of types $\Gamma = \{t_1, ..., t_j\}$, we train a multi-class text classifier to determine whether a snippet is the description of an entity of a given type. Before passing it to the classifier, the text of the snippet is converted to lower case and tokenized, each token corresponding to a word in the English dictionary; tokens that correspond to English stopwords are removed and the remaining are stemmed with the Porter algorithm [21]. Each token is associated with its normalized frequency in the snippet, that is obtained by dividing the number of its occurrences by the length of the snippet. The set of tokens, along with their relative frequencies, form the *features* used by the text classifier.

Given the content $T(i, j)$ of a cell, our algorithm retrieves the top-k snippets from *Bing* and, for each type $t \in \Gamma$, determines the number of snippets s_t that are classified as a description of an entity of type t; the type t_{max} such that $s_{t_{max}} > s_t$, for all $t \in \Gamma$, is selected as the type of the entity in $T(i, j)$ provided that $s_{t_{max}} > k/2$. In other words, the content of a cell $T(i, j)$ is considered as a reference to an entity of type $t \in \Gamma$ if and only if more than half of the snippets are classified as descriptions of an entity of type t; otherwise, $T(i, j)$ is not annotated. The annotation of cell $T(i, j)$ is assigned a score $S_{i,j}$ that expresses the probability that the content of $T(i, j)$ is a reference to an entity of type t. The score is determined as follows:

$$S_{i,j} = \frac{s_t}{k} \tag{1}$$

Classifier training.

Here we are confronted with the challenge of training a classifier to possibly handle a large number of entity types. To make our approach scalable, we need a method to create training and test sets that involves as little manual intervention as possible. More specifically, given a type $t \in \Gamma$ we need to collect snippets that describe entities of type t, referred to as *positive* examples. To this extent, the following procedure is used:

- A set P is created that contains entities of type t (*positive entities*). The entities are obtained from *DBpedia* [1].

- Up to 10 snippets are collected from *Bing* for each entity $e \in P$. The query submitted to *Bing* is a phrase obtained by concatenating the name of e and the name of t. For instance, if the name of e is "Melisse" and the name of t is "restaurant", the query submitted to *Bing* is "Melisse restaurant". The rationale of this choice is that the name of the type disambiguates the query so as to make sure to obtain snippets that describe the intended entity.

- Of the snippets obtained in the previous step, 75% are used to form the training set TR and 25% to form the test set TE.

In order to create the set P, we resort to the *categories* that *DBpedia* uses to organize entities based on their types. For example, the entity labelled "Musée du Louvre", which denotes the famous museum in Paris, France, belongs, among others, to the categories named "Art museums and galleries in Paris", "History museums in France" and "National museums of France". *DBpedia* organizes the categories in a hierarchy; thus, "History museums in France" is a subcategory of "Museums in France" which, in turn, is a subcategory of "Museums in Europe".

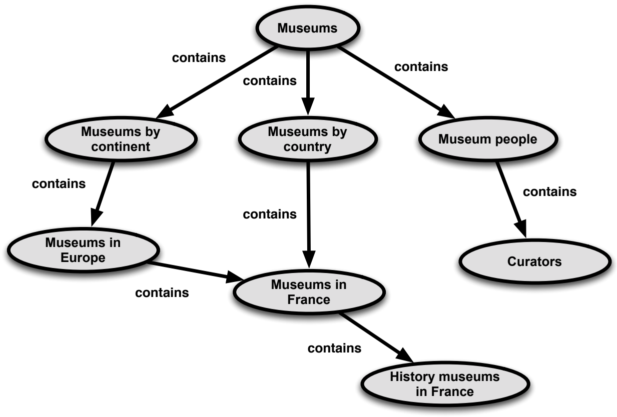


Figure 6: An excerpt of the category network in DBpedia rooted at category "Museums".

As shown in Figure 6, *DBpedia* categories are better represented as a graph, usually called the *category network*, where each node corresponds to a category and each edge describes a relation of containment between two categories.

To obtain the set P, we manually identify the category ρ (referred to as the *root*) that contains all the positive entities (e.g. "Museums"); next, we visit the category network to get all the subcategories of ρ, which is realized by iterating a SPARQL query on each subcategory of ρ. This way, we create a set C_{pos} of categories that contain positive entities; however, for how the categories are organized, it is still possible that C_{pos} includes some categories that contain negative entities. Referring to Figure 6, the category named "Curators" is a subcategory of "Museums", but it does not contain entities of type museum at all. To address this point, we use an heuristics that, although simple, considerably reduces, according to our observations, the number of categories with negative entities in C_{pos}; the heuristics consists of removing from C_{pos} all categories whose names do not contain the name of type t (e.g., *museum* in our example). Finally, the set P is created by randomly selecting entities that belong to any of the categories in C_{pos}.

One might argue at this point that the fact that we use *DBpedia* to create a training/test set contradicts our previous statement that we are not bound to any specific knowledge base or catalogue for annotating tables. However, we note that we do not use *DBpedia* for annotating entities but only for *training* our classifier, which will be then able to discover and annotate entities of a type t that do necessarily belong to *DBpedia*. More specifically, *DBpedia* does not contain a reference to, say, every restaurant around the world, but only a small subset. This subset is large enough to train a text classifier to determine whether a snippet is a description of a restaurant; thus, after training, the classifier is able to annotate an entity as a restaurant even if that entity is not in *DBpedia*. Nevertheless, it is true that if *DBpedia* does not provide any entity of a certain type t, we do not have any training data; however, we can still resort to another knowledge base that provides some entities of type t, such as *Yago* or *Probase*, which, when available, will provide entities of more than 2 million types. The reason why we selected *DBpedia* is that it provides training entities for most of the types that we need for our application (points of interest).

5.2.2 *Disambiguating Queries with Spatial Information*

For some types of entities, such as points of interest in cities or, in general, those that have a precise position in space, it is not uncommon to find spatial data in tables. For instance, if a table contains information on restaurants, it will most likely report their addresses. In GFT tables spatial information is usually clearly marked, as it occurs in columns with type LOCATION. In general tables the identification of spatial data is trickier and is an interesting research problem itself; in this paper we do not intend to solve it and we assume that one can use existing techniques such as the ones described by Borges et al. [2].

We can leverage these spatial data to disambiguate the query submitted to *Bing* to obtain the snippets. Referring to Figure 4, our algorithm may disambiguate the query for the term "Melisse" by appending the spatial information, such as the name of the city, that is found in the second column ("Santa Monica" in the example). As a result, the top-k snippets returned by *Bing* are likely to mostly refer to the restaurant named "Melisse" instead of the Jazz label with the same name.

One problem here is that spatial information in GFT tables often comes as postal addresses, which are difficult to parse because their format depends on the country. As a result, extracting information from an address, such as city/state/country name, is not straightforward. Besides, in many tables we came across, addresses are incomplete, and just report the street number and name and, possibly, the zip code. Therefore, we are confronted with the problem of obtaining the city name (and, possibly, the name of the state/country) from an address.

To this extent, we resort to online geocoding services such as the *Google Geocoding API* [6], which parses an address and breaks it down into different components, such as street, city, state and country, each identifying a geographic location. Such geographic locations are in a *containment* relationship defined as follows: streets are contained by cities, which are contained by states which in turn are contained by countries. Since the containment is a hierarchical relationship, any geographic location (e.g. a street) has a *direct* or *most specific* container (e.g. a city) and *indirect* or *less specific* containers (e.g. states and countries). Note that if the address is partial, the API can still retrieve the name of the city or cities to which the address may refer; therefore, we are left with the problem of resolving the ambiguities.

Let $T(i, j)$ be a cell containing an address for which the Google Geocoding API returns a set $L_{i,j}$ of geographic locations that might correspond to the address. For instance, in Figure 7a the address in cell $T(12, 1)$ may refer to a location in Baltimore, Maryland, or Washington, D.C. Resolving the ambiguity of the address amounts to choosing the location in $L_{i,j}$ that is the correct *interpretation* of the address in $T(i, j)$.

[6]`http://code.google.com/apis/maps/documentation/geocoding/`

We note that the i-th row may very well contain other cells with spatial values (e.g. "Washington, D.C.") that give further details on the partial address in cell $T(i, j)$ (e.g. "1600 Pennsylvania Avenue"). Similarly, other cells in the j-th column may have references to geographic locations (e.g. "College Park, GA") that have the same direct container as one of the locations in $L_{i,j}$ (e.g. "Washington, GA"), which therefore is more likely than the others to be the correct interpretation for $T(i, j)$.

In order to select the right interpretation for a cell $T(i, j)$ such that $|L_{i,j}| > 1$, we use a procedure that is close in spirit to *PageRank*. First, we create a graph with a node n_l for each geographic location $l \in L_{i,j}$, $1 \leq i \leq n$ and $1 \leq j \leq m$ (Figure 7b). For instance, in Figure 7b the location "Pennsylvania Avenue, Baltimore, MD, USA" is represented by a node labelled "12, 1 Pennsylvania Avenue, Baltimore, MD, USA", because it is an interpretation of the toponym in cell $T(12, 1)$ of Figure 7b. Note that any geographic location l that occurs in two distinct sets $L_{i,j}$ and $L_{h,k}$, where $h \neq i$ or $k \neq j$, is represented by two distinct nodes in the graph. For any pair of nodes n_{l_1}, n_{l_2}, such that $l_1 \in L_{i,j}$ and $l_2 \in L_{h,k}$, we add a directed edge from n_{l_1} to n_{l_2} if and only if the following two conditions are fulfilled:

1. $i = h$ or $j = k$ (that is, l_1 and l_2 are associated to cells that belong to the same row or column, but not to the same cell);
2. l_1 and l_2 share the same direct geographic container.

Referring to Figure 7b, there is an edge between node "12,1 Pennsylvania Ave, Washington, D.C" and "12,2 Washington, D.C., USA" because they are in the same row and share the same geographic container, that is "Washington, D.C.".

In practice, the presence of an edge from n_{l_1} to n_{l_2} means that $l_1 \in L_{i,j}$ chalks up a vote for $l_2 \in L_{h,k}$ to be the correct interpretation for the cell $T(h, k)$. Obviously, if n_{l_1} is the only location in $L_{i,j}$, which means that the address in $T(i, j)$ is unambiguous, the vote cast for l_2 has more importance than if l_1 was only one of the multiple possible interpretations for $T(i, j)$. Same considerations apply when many locations vote for l_1. We therefore need to associate each node n_l with a score $S(n_l)$ that is proportional to the number of locations that "vote" for l. To this extent, we use an algorithm that iteratively computes the scores of each node until the scores converge to a fixed point. Initially, for each node n_l, such that $l \in L_{i,j}$, we have

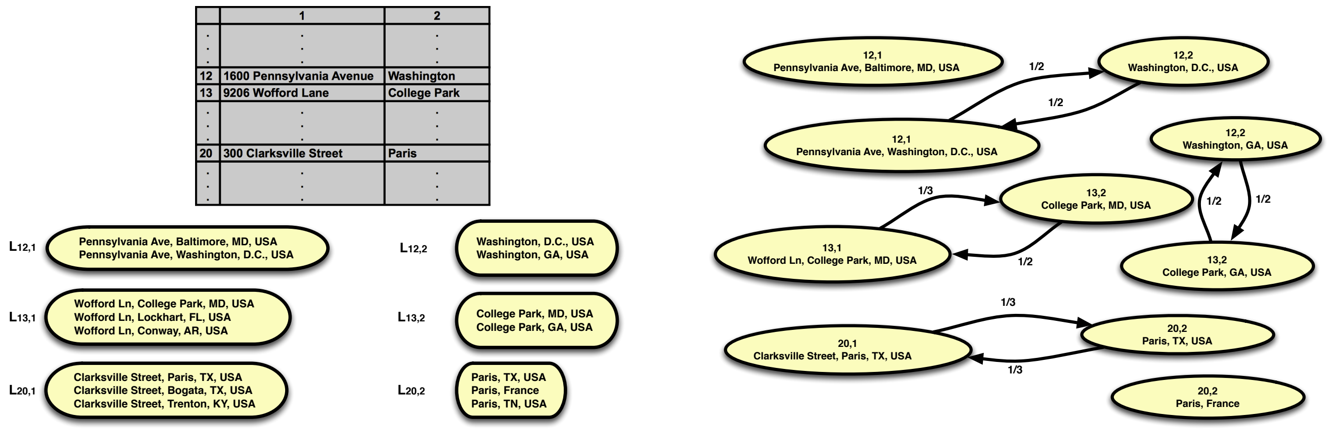
$$S(n_l) = \frac{1}{|L_{i,j}|}$$

In figure 7b the vote of node "12,1 Pennsylvania Ave., Washington, D.C., USA" to node "12,2 Washington, D.C., USA" is worth 1/2 at the first iteration, because the cell $T(12, 1)$, of which the first node represents one possible interpretation, has two possible interpretations. At each iteration, the score $S(n_l)$ is updated based on the following formula:

$$S(n_l) = \sum_{v \in IN(n_l)} S(v)$$

where $IN(n_l)$ is the set of nodes that link to n_l.

Finally, the correct interpretation for the cell $T(i, j)$ is the location l whose corresponding node n_l in the graph has the largest score across all nodes that correspond to locations


1
2
12 1600 Pennsylvania Avenue Washington
13 9206 Wofford Lane College Park
20 300 Clarksville Street Paris
L12,1
Pennsylvania Ave, Baltimore, MD, USA
Pennsylvania Ave, Washington, D.C., USA
L12,2
Washington, D.C., USA
Washington, GA, USA
L13,1
Wofford Ln, College Park, MD, USA
Wofford Ln, Lockhart, FL, USA
Wofford Ln, Conway, AR, USA
L13,2
College Park, MD, USA
College Park, GA, USA
L20,1
Clarksville Street, Paris, TX, USA
Clarksville Street, Bogata, TX, USA
Clarksville Street, Trenton, KY, USA
L20,2
Paris, TX, USA
Paris, France
Paris, TN, USA
12,1 Pennsylvania Ave, Baltimore, MD, USA
12,2 Washington, D.C., USA
1/2
1/2
12,1 Pennsylvania Ave, Washington, D.C., USA
12,2 Washington, GA, USA
1/3
13,2 College Park, MD, USA
1/2
1/2
13,1 Wofford Ln, College Park, MD, USA
1/2
13,2 College Park, GA, USA
1/3
20,2 Paris, TX, USA
20,1 Clarksville Street, Paris, TX, USA
1/3
20,2 Paris, France

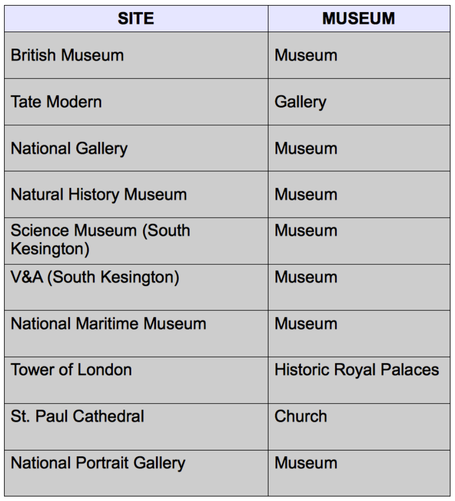
(a) Association toponyms - interpretations (b) Graph obtained from (a)

Figure 7: Disambiguating toponyms in tables.

in $L_{i,j}$. If the nodes corresponding to two or more locations in $L_{i,j}$ have the same score, we choose one randomly as the interpretation for $T(i, j)$.

5.3 Elimination of Spurious Annotations

In a table containing references to entities of type t, it is not uncommon to find cells that contain phrases that are related to the domain of those entities and they may be mistakenly annotated as being names of entities of type t. Consider the example of Figure 8, where the second column



SITE	MUSEUM
British Museum	Museum
Tate Modern	Gallery
National Gallery	Museum
Natural History Museum	Museum
Science Museum (South Kesington)	Museum
V&A (South Kesington)	Museum
National Maritime Museum	Museum
Tower of London	Historic Royal Palaces
St. Paul Cathedral	Church
National Portrait Gallery	Museum

Figure 8: Example of a column with repeated values that can be misclassified.

lists the types of the entities occurring in the first column. In particular, the term “Museum” is repeated in multiple cells, which, as expected, are classified as references to entities of type “Museum”, because the snippets returned in response to the query “Museum” look like those that describe entities of type “Museum”. No matter how good the text classifier is, we cannot realistically expect that it can capture such subtleties. We therefore exploit the column coherence principle to rule out spurious annotations that derive from the misclassification of snippets.

More specifically, we propose to combine the scores assigned by Equation 1 to the annotated cells in order to obtain a global score for each column. Ideally, the column with the highest score is the one that has references to entities of type t, whereas the others only happen to have ambiguous content that is misclassified. The rationale of the approach is that a column with many cells (with distinct values) having a high score is more likely to be the column that contains references to entities of type t than columns that have only few cells with (possibly) high score.

Let j be the index of a column. We define the global score S_j, which expresses the likelihood that column j contains entities of type t, as follows:

$$S_j = \sum_{i=1}^{n} \ln\left(\frac{1}{o_{i,j}} \cdot S_{i,j} + 1\right) \tag{2}$$

where $S_{i,j}$ is the score assigned to cell $T(i, j)$ by Equation 1 and $o_{i,j}$ is the number of occurrences of the content of cell $T(i, j)$ across the whole column j. The factor $1/o_{i,j}$ is introduced to diminish the effect of high scores assigned to cells whose content is repeated in many cells (such as the ones containing “Museum” in Figure 8).

6. EVALUATION

In this section we present the evaluation of our algorithm, which is composed of four parts.

We first detail in Section 6.1 the characteristics of the training/test sets obtained through the procedure presented in Section 5.2.1 and we evaluate two text classifiers, one based on a support vector machine (SVM) and the other on Naive Bayes classification techniques.

In Section 6.2, we discuss the evaluation results obtained by running our algorithm on a set of 40 tables extracted from GFT to annotate entities of 12 different types belonging to three different categories:

1. Points of interest of cities: RESTAURANTS, MUSEUMS, THEATRES, HOTELS, SCHOOLS, UNIVERSITIES and MINES;
2. People: ACTORS, SCIENTISTS, SINGERS;
3. Cinema: FILMS and SIMPSON’S EPISODES.

	SVM			Bayes			TIN			TIS		
Type	P	R	F	P	R	F	P	R	F	P	R	F
RESTAURANTS	0.89	0.69	**0.78**	0.59	0.80	0.68	0.64	0.1	0.17	1.0	0.19	0.32
MUSEUMS	0.83	0.82	**0.82**	0.45	0.93	0.61	1.0	0.61	0.76	1.0	0.59	0.74
THEATRES	0.83	0.76	**0.79**	0.34	0.89	0.49	1.0	0.18	0.31	1.0	0.29	0.45
HOTELS	0.74	0.89	**0.81**	0.23	0.92	0.37	0.15	0.07	0.10	0.96	0.60	0.74
SCHOOLS	0.96	0.91	**0.93**	0.75	0.96	0.84	0.78	0.56	0.65	0.98	0.88	0.93
UNIVERSITIES	0.94	1.0	**0.97**	0.88	1.0	0.94	0.0	0.0	0.0	0.98	0.88	0.93
MINES	1.0	1.0	**1.0**	0.75	0.96	0.84	0.0	0.0	0.0	1.0	0.12	0.21
AVERAGE	0.88	0.87	**0.87**	0.57	0.92	0.68	0.51	0.22	0.28	0.99	0.51	0.62
ACTORS	0.90	0.71	**0.79**	0.50	0.93	0.65	0.0	0.0	0.0	1.0	0.13	0.23
SINGERS	0.69	0.75	**0.72**	0.09	0.73	0.16	0.0	0.0	0.0	0.0	0.0	0.0
SCIENTISTS	0.80	0.70	**0.75**	0.36	0.91	0.52	0.0	0.0	0.0	0.0	0.0	0.0
AVERAGE	0.80	0.72	**0.75**	0.32	0.86	0.44	0.0	0.0	0.0	0.33	0.04	0.08
FILMS	0.80	0.75	**0.77**	0.44	0.65	0.52	0.0	0.0	0.0	0.0	0.0	0.0
SIMPSON'S EPISODES	0.83	0.91	**0.87**	0.65	0.87	0.74	0.0	0.0	0.0	0.0	0.0	0.0
AVERAGE	0.82	0.83	**0.82**	0.55	0.76	0.63	0.0	0.0	0.0	0.0	0.0	0.0

Table 1: Evaluation of the algorithm.

The entities in the first category are of direct interest to our application (see Section 1). Moreover, except MINES, they all have spatial information that we used to evaluate the query disambiguation procedure described in Section 5.2.2. We observed that overall the names of the points of interest that we found in the tables did not pose many problems of ambiguity. This can be explained by the fact that points of interest have long names that tend to be less ambiguous than short ones. Since our intent was to assess the robustness of our algorithm to ambiguity, we chose to consider tables containing information on people, whose names tend to be highly ambiguous. As for the third category, we picked one that does not fall into either of the previous ones and is well represented in GFT tables. Note that we selected on purpose some types that are linked by a subsumption relation; more specifically, UNIVERSITIES and SIMPSON'S EPISODES are respectively subtypes of SCHOOLS and FILMS. This is to evaluate the ability of our algorithm to determine the correct fine-grained type of an entity even when confronted with snippets that describe similar entities.

In Section 6.3 we compare our algorithm against the one proposed by Limaye et al. [14], which for brevity we term LIMAYE, after the name of its first author. To the best of our knowledge, this is the only method for which the dataset used for the evaluation is publicly available, which allows us the comparison.

Finally, in Section 6.4 we present some considerations on the efficiency of our method in terms of computational time and we discuss its scalability.

All experiments have been conducted on a desktop computer running *Ubuntu* 11.04 with a Intel Core i7-2600 3.40 Ghz and 8 GB of RAM.

6.1 Setup of the Text Classifiers

For each of the 12 types under evaluation, we obtained the snippets from *DBpedia* with the procedure described in Section 5.2 in order to form the training and test sets. On average, the creation of the training/test corpora took around 2 hours, due to the latency of querying *DBpedia* and *Bing*. Table 2 shows that for each type the training (TR) and the testing (TE) sets include respectively up to 40,000 and 13,000 snippets with some exceptions such as SIMPSON'S EPISODES, for which *DBpedia* provides few entities.

Type	**|TR|**	**|TE|**	BAYES	SVM
RESTAURANTS	37,231	12,369	0.95	**0.97**
MUSEUMS	44,220	14,683	0.93	**0.97**
THEATRES	41,762	13,892	0.94	**0.98**
HOTELS	44,640	14,822	0.96	**0.98**
SCHOOLS	44,205	14,768	0.96	**0.99**
UNIVERSITIES	44,881	14,979	0.97	**1.0**
MINES	17,044	5,743	0.91	**0.94**
ACTORS	44,837	14,944	0.93	**0.97**
SINGERS	44,829	14,935	0.92	**0.95**
SCIENTISTS	44,867	14,945	0.92	**0.98**
FILMS	44,886	14,971	0.95	**0.99**
SIMPSON'S EPISODES	7,318	2,462	0.95	**0.97**

Table 2: Results of the training/test phase.

We trained and tested two text classifiers, a Support Vector Machine (SVM) and a Naive Bayes classifier (BAYES). For the first we opted for a C-SVC based on the implementation provided by LibSVM [7]. We trained the C-SVC with a RBF kernel and we followed the grid-search procedure with 10-fold cross validation described in [13] to select the optimal values of the parameter *cost* of the C-SVC and the parameter γ of the kernel, both set to 8. For BAYES, we

[7]`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

used the implementation provided by LingPipe [8]; we turned off length normalization and set the prior counts to 1.0. The fourth and fifth column of Table 2 show the F-measure obtained while testing SVM and BAYES respectively; as we expected, SVM outperformed BAYES across all types.

6.2 Evaluation of the Algorithm

We manually obtained 40 tables from GFT containing references to entities of the twelve selected types. In total we have 287 references to restaurants, 240 to museums, 160 to theatres, 67 to hotels, 109 to schools, 150 to universities, 30 to mines, 50 to actors, 120 to singers, 100 to scientists, 24 to films and 34 to episodes of the Simpson's. Each table was manually annotated by one person, so as to have a gold standard against which we compared our algorithm.

For each type, we evaluated our algorithm by using SVM and BAYES; at this point we did not use the disambiguation procedure described in Section 5.2.2. We also evaluated two baselines that we termed TYPEINNAME (TIN) and TYPEINSNIPPET (TIS):

- TIN annotates a cell $T(i,j)$ with type t, and sets the score $S_{i,j}$ to 1.0 only if $T(i,j)$ contains the name of type t (e.g. "restaurant"); otherwise, $T(i,j)$ is annotated as a negative example (that is not a reference to an entity of type t) and $S_{i,j} = 0$.

- TIS annotates a cell $T(i,j)$ with type t if the majority of the snippets retrieved by querying *Bing* contains the name of type t. The score $S_{i,j}$ is set as in Equation 1.

For each type t, we evaluated the methods by using *precision* (P), *recall* (R) and *f-measure* (F), defined as follows:

$$P = \frac{C_t}{A_t} \quad R = \frac{C_t}{T_t} \quad F = 2 \cdot \frac{P \cdot R}{P + R}$$

where:

- C_t is the set of entities that the algorithm correctly annotates with type t;

- A_t is the set of entities of type t for which the algorithm determines an annotation.

- T_t is the set of all entities of type t.

The results are shown in Table 1. The evaluation shows that our algorithm coupled with SVM consistently outperforms the other methods across all types. This comes as no surprise, as SVMs have proved to be superior in many classification tasks, including text classification.

It is interesting to note that while the use of SVM favours precision, the use of BAYES favours recall; however, while SVM leads to a good balance between precision and recall, which is reflected in the high values of the f-measure, the precision obtained by BAYES is very poor, with few exceptions. A possible explanation to this is that each snippet is a very short text, and therefore it is described by few features; apparently, SVM is more robust than BAYES when it comes to classifying texts with few features.

Our algorithm (coupled with SVM) shows good results even while annotating tables containing references to entities whose names are often ambiguous (ACTORS, SINGERS and SCIENTISTS). Compared to the results obtained on the points of interest, we observe a slight drop in the average precision (0.88 versus 0.80) and a significant drop in the average recall (0.87 versus 0.72), which is an indication that the classifier finds difficult to determine whether an entity is a person from a set of snippets. However, the overall accuracy remains good (0.75).

We did not experience particular problems with types that are linked by a subsumption relation (SCHOOLS, UNIVERSITIES and FILMS, SIMPSON'S EPISODES); in both cases, the accuracy of the algorithm is high.

As for the baseline methods, it is interesting to note that, although they are straightforward, they can be quite effective in some cases; the high precision achieved by TIS on the points of interest is an evidence to this. However, both baselines are conservative in the sense that they annotate entities only when they are reasonably sure that they are positive entities. As a result, they often show high precision and very low recall. The two baselines are definitely effective for entities such as museums and restaurants, whose names and snippets tend to contain an indication of their type (e.g. "Louvre Museum"); but overall, they don't have stable results across all the types, which makes them unsuitable for a large scale annotation of tables.

Type	**SVM**	**SVM+ postproc.**	**SVM+ postproc.+ disambig.**
RESTAURANTS	0.71	0.78	**0.81**
MUSEUMS	0.58	0.83	**0.83**
THEATRES	0.74	0.80	**0.83**
HOTELS	0.65	0.81	**0.87**
SCHOOLS	0.75	0.94	**0.94**
UNIVERSITIES	0.97	0.97	**0.98**
MINES	0.62	**1.0**	–
ACTORS	0.79	**0.80**	–
SINGERS	0.51	**0.72**	–
SCIENTISTS	0.68	**0.75**	–
FILMS	0.63	**0.77**	–
SIMPSON'S EPISODES	0.85	**0.87**	–

Table 3: Evaluation of the algorithm without postprocessing (second column), with postprocessing (third column) and with postprocessing and disambiguation (fourth column).

We conclude our discussion by showing the contribution given to the algorithm by the elimination of the spurious annotations, which represents the post-processing step, and the disambiguation procedure described in Section 5.2.2. To this extent, we run our algorithm coupled with SVM based on three different settings:

- Without using post-processing nor disambiguation (SVM);

- Using the post-processing but not the disambiguation (SVM + postproc.). This is the setting we used to obtain the results shown in Table 1;

[8]`http://alias-i.com/lingpipe/`

- Using the post-processing and the disambiguation (SVM + postproc. + disambig.).

Table 3 shows the f-measure obtained for each setting. The post-processing increases dramatically the accuracy of the algorithm, which proves the validity of the scoring mechanism that we described in Section 5.3. One might argue that since the algorithm without post-processing does not provide a high accuracy, the SVM does not effectively determine the types of the entities based on the snippets; however, we must note that the scoring mechanism used by the post-processing is based on the scores that are computed by the SVM.

As for the disambiguation method, we used it only on the points of interest (except mines), because they are the only entities for which we found spatial data in the tables. As expected, disambiguating the queries leads to a better accuracy, although the improvement is not dramatic, which might be due to the fact that the names of the points of interest that occur in our tables are not highly ambiguous. Probably, if we had a way to disambiguate the queries relative to names of people, we would notice a more remarkable improvement. We did not investigate further this aspect, as our application required us to collect information on points of interest, for which the disambiguation method described in this paper works well.

6.3 Comparison

We run our algorithm (with the setting SVM + postproc.) on the same dataset used to evaluate LIMAYE [14] in order to compare the results. This dataset, called *Wiki_Manual*, includes 36 tables obtained from Wikipedia articles which mostly contain entities of the types used in our evaluation. We obtained a f-measure of 0.84, which is comparable to the one (0.8382) reported for LIMAYE as the entity annotation accuracy. Unfortunately, no precision nor recall is given for LIMAYE, which would allow a more complete comparison.

This comparison supports our statement that our algorithm can complement existing approaches in entity annotation. Indeed, besides achieving an accuracy that is comparable to one of the main state-of-art approaches, our algorithm is also able to discover new entities, that do not necessarily belong to a local catalogue.

6.4 Efficiency

The running time of our algorithm is dominated by the latency time required to connect to the search engine and, while using the query disambiguation method, to the Google Geocoding service. We observed that on average the algorithm takes 0.5 seconds to process a row; this means that our algorithm is quite effective while handling small tables having up to 500 rows. Most of the tables we came across in GFT have a size that respects this limit; in particular, the average number of rows in the tables in our datasets is 50.

As we anticipated earlier, we can combine our algorithm with existing approaches to make the annotation of large tables more efficient. More specifically, we may use LIMAYE to annotate entities that belong to a pre-compiled catalogue, and resort to the search engine only to annotate previously unseen entities. Since in general we expect a table to have a combination of known and unknown entities, this should bring down the running time of the annotation. We will explore this point as a future work.

As for the scalability, we remark that our algorithm is fully automatic except for the selection of the category in *DBpedia* that best represents a type of entities. This is not a big issue if entities that we want to annotate in tables belong to a limited number of types, which is the case of our application, for which we identified up to 100 types. However, if we intended to use our algorithm for annotating entities of any type in Probase, which includes up to two million types, we would need a way to automatically select the category that best represents a type. Again, this point is an interesting direction for future work.

7. CONCLUDING REMARKS

In this paper we described an algorithm that annotates entities in tables that is part of an application for creating a repository of information on points of interest of cities extracted from Google Fusion Tables; this repository is used by a Web application that allows a faceted browsing over the data. Our algorithm complements existing approaches, because it can discover and annotate entities that do not necessarily belong to a pre-compiled catalogue. Our evaluation shows that the proposed algorithm achieves an accuracy that is comparable to that reported for one of the major state-of-art algorithm. As future work, we intend to integrate our algorithm with one of the existing approaches in order to obtain an efficient and reliable tool for entity annotation in table. More importantly, we intend to improve the scalability of our algorithm so as to be able to annotate entities of several thousand types of entities.

8. REFERENCES

[1] BIZER, C., LEHMANN, J., KOBILAROV, G., AUER, S., BECKER, C., CYGANIAK, R., AND HELLMANN, S. DBpedia - A Crystallization Point for the Web of Data. *Web Semant. 7* (September 2009), 154–165.

[2] BORGES, K. A. V., LAENDER, A. H. F., MEDEIROS, C. B., AND DAVIS, JR., C. A. Discovering Geographic Locations in Web Pages Using Urban Addresses. In *Proceedings of the 4th ACM Workshop on Geographical Information Retrieval* (New York, NY, USA, 2007), GIR '07, ACM, pp. 31–36.

[3] CAFARELLA, M. J., HALEVY, A., WANG, D. Z., WU, E., AND ZHANG, Y. WebTables: Exploring the Power of Tables on the Web. *Proc. VLDB Endow. 1*, 1 (2008), 538–549.

[4] CIMIANO, P., AND VÖLKER, J. Towards Large-scale, Open-domain and Ontology-based Named Entity Classification. In *In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP'05)* (2005), INCOMA Ltd, pp. 166–172.

[5] DOAN, A., RAMAKRISHNAN, R., AND HALEVY, A. Y. Crowdsourcing Systems on the World-Wide Web. *Commun. ACM 54* (2011), 86–96.

[6] FLEISCHMAN, M., AND HOVY, E. Fine Grained Classification of Named Entities. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1* (Stroudsburg, PA, USA, 2002), COLING '02, Association for Computational Linguistics, pp. 1–7.

[7] GANTI, V., KÖNIG, A. C., AND VERNICA, R. Entity Categorization over Large Document Collections. In

Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (New York, NY, USA, 2008), KDD '08, ACM, pp. 274–282.

[8] Giuliano, C. Fine-grained Classification of Named Entities Exploiting Latent Semantic Kernels. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning* (Stroudsburg, PA, USA, 2009), CoNLL '09, Association for Computational Linguistics, pp. 201–209.

[9] Gonzalez, H., Halevy, A. Y., Jensen, C. S., Langen, A., Madhavan, J., Shapley, R., Shen, W., and Goldberg-Kidon, J. Google Fusion Tables: Web-centered Data Management and Collaboration. In *Proceedings of the 2010 International Conference on Management of Data* (New York, NY, USA, 2010), SIGMOD '10, ACM, pp. 1061–1066.

[10] Guo, X., Chen, Y., Chen, J., and Du, X. ITEM: Extract and Integrate Entities from Tabular Data to RDF Knowledge Base. In *Proceedings of the 13th Asia-Pacific Web Conference on Web Technologies and Applications* (Berlin, Heidelberg, 2011), APWeb'11, Springer-Verlag, pp. 400–411.

[11] Han, L., Finin, T., Parr, C., Sachs, J., and Joshi, A. RDF123: From Spreadsheets to RDF. In *The Semantic Web - ISWC 2008*, vol. 5318 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 2008, pp. 451–466.

[12] Hignette, G., Buche, P., Dibie-Barthélemy, J., and Haemmerlé, O. Fuzzy Annotation of Web Data Tables Driven by a Domain Ontology. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications* (Berlin, Heidelberg, 2009), ESWC 2009, Springer-Verlag, pp. 638–653.

[13] Hsu, C. W., Chang, C. C., and Lin, C. J. A Practical Guide to Support Vector Classification. Retrieved online at `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`, 2003.

[14] Limaye, G., Sarawagi, S., and Chakrabarti, S. Annotating and Searching Web Tables Using Entities, Types and Relationships. *Proc. VLDB Endow. 3* (September 2010), 1338–1347.

[15] Mulwad, V. DC proposal: Graphical Models and Probabilistic Reasoning for Generating Linked Data from Tables. In *Proceedings of the 10th international conference on The semantic web - Volume Part II* (Berlin, Heidelberg, 2011), ISWC'11, Springer-Verlag, pp. 317–324.

[16] Ni, Y., Zhang, L., Qiu, Z., and Wang, C. Enhancing the Open-domain Classification of Named Entity Using Linked Open Data. In *Proceedings of the 9th International Semantic Web Conference on The semantic Web - Volume Part I* (Berlin, Heidelberg, 2010), ISWC'10, Springer-Verlag, pp. 566–581.

[17] Quercini, G., Setz, J., Sonntag, D., and Reynaud, C. Facetted Browsing of Extracted Fusion Tables Data for Digital Cities. In *Proceedings of the Web of Linked Entities Workshop in conjunction with the 11th International Semantic Web Conference (ISWC 2012)* (2012), pp. 94 – 105.

[18] Setz, J., Quercini, G., Sonntag, D., and Reynaud, C. Facetted Search on Extracted Fusion Tables Data for Digital Cities. In *35th Annual German Conference on Artificial Intelligence (Demo paper)* (2012).

[19] Suchanek, F. M., Kasneci, G., and Weikum, G. Yago: a Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web* (New York, NY, USA, 2007), WWW '07, ACM, pp. 697–706.

[20] van Assem, M., Rijgersberg, H., Wigham, M., and Top, J. Converting and Annotating Quantitative Data Tables. In *Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part I* (Berlin, Heidelberg, 2010), ISWC'10, Springer-Verlag, pp. 16–31.

[21] van rijsbergen, C. J., Robertson, S. E., and Porter, M. F. New models in probabilistic information retrieval. In *British Library Research and Development Report, no. 5587* (1980), London: British Library.

[22] Venetis, P., Halevy, A., Madhavan, J., Paşca, M., Shen, W., Wu, F., Miao, G., and Wu, C. Recovering Semantics of Tables on the Web. *Proc. VLDB Endow. 4* (2011), 528–538.

[23] Wang, J., Wang, H., Wang, Z., and Zhu, K. Q. Understanding Tables on the Web. In *Conceptual Modeling - 31st International Conference ER 2012* (2012), vol. 7532 of *Lecture Notes in Computer Science*, Springer, pp. 141–155.

[24] Wu, W., Li, H., Wang, H., and Zhu, K. Q. Probase: a Probabilistic Taxonomy for Text Understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2012), SIGMOD '12, ACM, pp. 481–492.

[25] Zicari, R. V. Google Fusion Tables. Interview with Alon Y. Halevy. http://www.odbms.org/blog/2011/08/google-fusion-tables-interview-with-alon-y-halevy/, 2011.