Towards Searching Amongst Tables

Paul Thomas
CSIRO,
Australia
paul.thomas@csiro.au

Rollin Omari*
The Australian National
University, Australia
u5598038@anu.edu.au

Tom Rowlands*

tom.rowlands@ieee.org

ABSTRACT

An increasing number of data sets are being published online, in institutional or government repositories as well as by individual researchers, journalists, and others. These data are often represented as tables of various kinds: however, repositories have poor search over and inside tables. It is difficult for a user to tell from a repository's portal whether a useful dataset is available, and this problem is only likely to get worse.

We describe this problem, and demonstrate that the naïve approach of full-text search is not appropriate. We describe an alternative, based on inferring types of data and indexing columns as a unit, and demonstrate some improvements in early success especially when long captions are not available.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information search and retrieval—*search process*.

1. SEARCHING AMONGST TABLES

A large, and growing, number of data sets are published on the web. Recently this has been driven not only by governments opening their data and putting it online but also by an increasing tendency to publish research data in various forms.

Formats for these data sets vary, but an obvious format, especially for data intended for a wide audience, is a simple two-dimensional table such as a spreadsheet. These are common: Google estimates 25,000 CSV files plus 54,000 Excel spreadsheets in `gov.au`, at early September 2015; and 25,000 plus 170,000 in `gov.uk`, numbers which exclude research institutions as well as data hidden behind search forms. These can also be important: for example, tables published in Australia include fundamental social and economic data as well as climate projections and other data of public interest.

Access to data sets is typically through specialised portals, including those from government (for example `data.gov.au`; `data.nsw.gov.au` and equivalents in other states; `data.govt.nz`; `data.govt.uk`; `data.gov`) and research (e.g. `data.csiro.au`, `ands.org.au`) as well as brokers (e.g. `govpond.org`). However, discovery tools at these portals remain limited and are often based on full-text searches of descriptions and other metadata.

Figure 1 gives examples of searching for ⟨ unemployment rate Australia 1990s ⟩ at two portals. In neither case are the results useful: the top results are for the wrong years and other results vary from off-topic to incomprehensible. (The search algorithms are unknown, but appear to be weighting by coordination level.) Further, the result pages do not summarise the data with regard to the query—there is no equivalent of the query-biased summary—which makes the search harder to understand and debug, and which means an entire data set must be downloaded to answer simple questions or even to make a judgement on its possible utility. Manually curating and marking up data for date range, geographic coverage, and so forth could give better results but would scale poorly.

Consider again the unemployment query. Table 1 gives four example tables, with fictitious data. A naïve strategy would be simply to index the contents of each table as if text, and use an established search engine to rank the tables. This would retrieve the most useful table (1a), but would behave poorly in the other cases. Table 1b does not mention "Australia" at all, although these are Australian placenames; the table is partially relevant but will not be returned, although query expansion may help. Table 1c illustrates a second class of error: the strings "1990", "1991", ... are all present but are scattered across different columns, and the naïve approach will not require co-located terms. Even if this requirement were added, Table 1d would still produce an error: here, the strings "1990", "1991", ... are all present in the same column but are instances of postcodes, not dates.

2. AN APPROACH

Ideally, it would be possible to match tables to queries without manual markup and without resorting to naïve fulltext indexing.

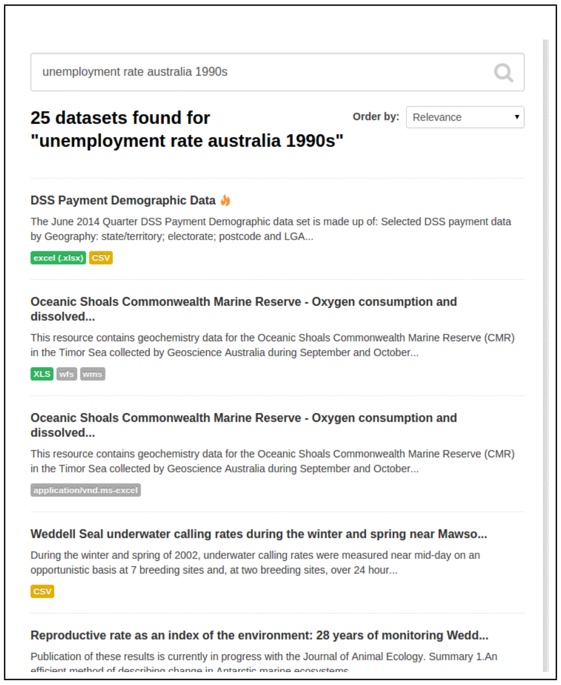
Examining our query ⟨ unemployment rate Australia 1990s ⟩ gives some hints towards an algorithm. What would a responsive document—a useful table—look like? Clearly, we are looking for unemployment figures. We can also observe that data should be from Australia; but that (for example) data from just New South Wales, or Canberra, is partially relevant or probably relevant. We also observe that we are looking for data from the period 1990–9; ideally for all years, but again possibly accepting data for just some.

This suggests we are looking for a table which contains at least two columns:

1. A column of numbers, called "unemployment rate", or some equivalent ("unemployment", "employment"), and with numeric values; equivalently, this column may have a generic name (or no name) and the table's description or title may include this key phrase.

*This work was carried out at CSIRO.

ADCS'15 December 8–9, 2015, Parramatta, NSW, Australia
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-4040-3/15/12 ... $15.00.
http://dx.doi.org/10.1145/2838931.2838941 .



unemployment rate australia 1990s

25 datasets found for "unemployment rate australia 1990s"

Order by: Relevance

DSS Payment Demographic Data

The June 2014 Quarter DSS Payment Demographic data set is made up of: Selected DSS payment data by Geography: state/territory; electorate; postcode and LGA...

excel (.xlsx) CSV

Oceanic Shoals Commonwealth Marine Reserve - Oxygen consumption and dissolved...

This resource contains geochemistry data for the Oceanic Shoals Commonwealth Marine Reserve (CMR) in the Timor Sea collected by Geoscience Australia during September and October...

XLS wfs wms

Oceanic Shoals Commonwealth Marine Reserve - Oxygen consumption and dissolved...

This resource contains geochemistry data for the Oceanic Shoals Commonwealth Marine Reserve (CMR) in the Timor Sea collected by Geoscience Australia during September and October...

application/vnd.ms-excel

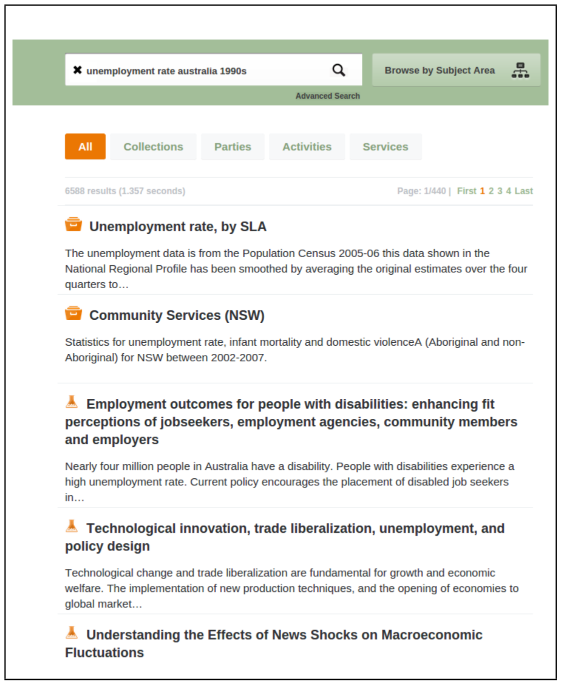
Weddell Seal underwater calling rates during the winter and spring near Mawso...

During the winter and spring of 2002, underwater calling rates were measured near mid-day on an opportunistic basis at 7 breeding sites and, at two breeding sites, over 24 hour...

CSV

Reproductive rate as an index of the environment: 28 years of monitoring Wedd...

Publication of these results is currently in progress with the Journal of Animal Ecology. Summary 1.An efficient method of describing change in Antarctic marine ecosystems



unemployment rate australia 1990s

6588 results (1.357 seconds) Page: 1/440 | First 1 2 3 4 Last

Unemployment rate, by SLA

The unemployment data is from the Population Census 2005-06 this data shown in the National Regional Profile has been smoothed by averaging the original estimates over the four quarters to…

Community Services (NSW)

Statistics for unemployment rate, infant mortality and domestic violenceA (Aboriginal and non-Aboriginal) for NSW between 2002-2007.

Employment outcomes for people with disabilities: enhancing fit perceptions of jobseekers, employment agencies, community members and employers

Nearly four million people in Australia have a disability. People with disabilities experience a high unemployment rate. Current policy encourages the placement of disabled job seekers in…

Technological innovation, trade liberalization, unemployment, and policy design

Technological change and trade liberalization are fundamental for growth and economic welfare. The implementation of new production techniques, and the opening of economies to global market…

Understanding the Effects of News Shocks on Macroeconomic Fluctuations

Figure 1: Example searches over existing data repositories. Keyword matching produces results which are either obviously irrelevant or simply inexplicable; and summaries are not tailored to the query.

2. A column of dates; most likely called for example “date” or “year”, and with values in the range 1990–1999.

Such a table might also have, for example, a column of placenames (inside or outside Australia; if outside, we should ignore it); and/or any number of other columns, which we are not interested in here and which do not bear on the table’s relevance.

In this example, the notions of *type* and of *columnar alignment* are critical. We make use of these ideas in two ways: first, when indexing tables; and second, when parsing user queries and rewriting them for retrieval.

2.1 Indexing tables

Tables are indexed column-wise, with each column having a uniform type and possibly a single header.

Types are selected to cover the variety of data in our sample, and a simple type inference algorithm determines the most likely type of each cell and therefore the most likely type of each column. Each possible type includes a representation for indexing, which determines how cells are indexed; and any number of “parents” type, more restrictive type with compatible representations. For example, the year type has the numeric type as a parent: all representations of years are also valid representations of numbers. These parent/child relationships define a lattice. The current implementation recognises six types, although it is easy to extend to others:

Strings are indexed as-is.

Numbers are parsed with optional suffixes (e.g. “M”, “k”, “%”), and are indexed with the precision given in the input. Currency values are indexed as numbers, again with the precision given in the input.

Boolean values can appear either as strings (in which case the parent type is string) or as 1/0 (in which case the parent type is integer), and are indexed as “True” or “False”.

Dates are indexed as a single token according to the precision in the table; e.g. “1990” is translated to “1990”, and “5 June 2016” to “20160605”. Financial years in the form “1990–91” or “1990–1991” are also recognised.

Placenames are recognised from a gazeteer and indexed as a single token which includes all enclosing regions, e.g. “NSW” is indexed as “australia_newsouthwales”, and “Sydney” is indexed as “australia_newsouthwales_sydney”.

Missing values are not indexed, but are treated specially when deciding the type of a column.

Inference runs for each column separately[1]. For each column, we look at the first 20 values and find the type for each. We then walk the lattice of types to find the first common ancestor—the most restrictive type which allows all values. For example, given `(1990, 12.3k, 12)` we infer (year, number, number) and select number; it is the “closest” type which covers all the values. For `(Sydney, Melbourne, television)` we select the string type. Missing values are allowed to be of any type, so `(10, NA, 11)` or `(12, , 13)` will each be treated as a column of numbers.

If the first value appears to be a different type to the rest, we flag it as a potential header. If any of the columns appear to have a header, they are all assumed to have one; otherwise none are. We also allow an optional header for the entire table.

Each column is then indexed as a separate extent, tagged with the appropriate type. Figure 2 illustrates the result for Table 1a.

2.2 Query parsing and rewriting

The query parser attempts to translate short, free-text, queries into a description of the columns and (optionally) cells which match. This uses the same type inference and tagging machinery as above,

[1]Code for the type inferrer, and the other parts of the table indexer and ranker, is available from the first author.

Australian unemployment		
Year	Level	Notes
1990	9%	...
1991	12%	...
⋮	⋮	⋮
1999	7%	...

(a) With additional data, and description in the title

Where	When	Unemployment
NSW	1990	9.00%
NSW	1991	11.65%
⋮	⋮	⋮
NSW	1999	6.87%

(b) Not for all Australia

State	Date	Unemployment	Investment
NSW	1990	9.0%	$1999
Qld	2005	5.6%	$1990
⋮	⋮	⋮	⋮
All Australia	1984	6.2%	$1993

(c) With "1990s" in more than one column

Australian unemployment	
Postcode	Level
1990	8%
1992	7%
⋮	⋮
2602	4%

(d) Apparent dates, but of the wrong data type

Table 1: Four candidate tables for the query ⟨ unemployment rate Australia 1990s ⟩. Naïvely indexing the tables as text will miss Table 1b—which is relevant—but return the barely-relevant Table 1c and non-relevant Table 1d. (Data is fictitious.)

except that instead of the indexing representation each query term is re-written using a query representation; each rewritten term is also annotated with the type of column it must appear in.

Query representations are generally the same as index representations, except that placenames and dates are treated as prefixes: that is, the query representation of ⟨ Australia ⟩ is "australia*" as this will match any Australian place in the index, and the query representation of ⟨ March 2014 ⟩ is "201403*" as this will match any date or time in that month. We also allow a small number of extra types, to deal with ranges such as ⟨ 1990s ⟩, which are unlikely to appear in the tables being searched.

To continue the example, given the query ⟨ unemployment rate Australia 1990s ⟩, type inference gives ⟨ unemployment$^{\text{string}}$ rate$^{\text{string}}$ Australia$^{\text{placename}}$ 1990s$^{\text{decade}}$ ⟩. Using the query representation appropriate to each type, and marking the acceptable columns for each term, gives a query specifying that:

- "unemployment" should occur inside a `string` or `header` extent;
- "rate" should occur in a `string` or `header`;
- "Australia*" should occur in a `placename` or `header`;
- "199*" should occur in a `date`.

This query is expressed as a combination of Indri "#or", "#combine", and field operators [6] (Figure 2) and run against the index built above.

Note that this rewritten query, by respecting type information and columnar layout, avoids the pitfalls of the naïve approach: Table 1b will match since "NSW" is indexed as "australia_newsouthwales"; Table 1c will be a poor match since only "1990" appears in the column of dates; and Table 1d will not match since the values "1990", etc, appear in a column of integers rather than a column of dates.

3. EARLY EVALUATION

We have begun evaluating these ideas by building a small test collection. A set of 35 tables and captions was collected from the website of the Australian Bureau of Statistics[2], and minimally reformatted for example by removing logos and links, and by collapsing multiple tabs to a single table. The resulting tables were indexed as plain text, and as typed columns as described above, using the Indri toolkit [6] in each case, both with and without the table caption.

Five colleagues—not authors of this paper—were asked to look at the tables and captions, think of information found in them, and write queries as if for a search engine. This gave us a set of 60 "known-item" queries, covering 26 of the 35 tables. Queries were run through Indri, either as-is for the full-text approach or after inference and rewriting for the proposed approach.

Effectiveness measures for the two methods are summarised in Table 2. We see small improvements in early success and in mean reciprocal rank, when table captions are included. The relatively good performance of the simple approach is possibly an artefact of our query generation, since participants had access to captions and often chose to copy parts of these. Where target tables are not known in advance, or where metadata is unavailable or not useful, we expect a larger gain: simulating this by removing captions, this is what we observe.

Examining queries shows no particular pattern of queries where our approach is poor, although there is certainly room for improvement. As expected, our approach does relatively well where a range of cells is described (e.g. "australia states government taxation 2000s", which implies a range of places and a range of dates).

4. RELATED WORK

Tables in text have received a good deal of attention: the survey of Embley et al. [2] for example discusses extracting tables from HTML and text; information extraction; deriving ontologies; subsetting, querying, and converting tables; and other table manipulations; over 98 references. Despite this, there is very little work on finding relevant tables in the first place, be they extracted from text or (as in our case) stand-alone.

A series of question-answering systems from Croft and colleagues [4, 5, 7] have considered table ranking as well as table extraction.

[2]See for example `http://www.abs.gov.au/websitedbs/CaSHome.nsf/Home/Economy+Datasets`.

```
Australian unemployment
Year   Level     Notes
----------------------
1990    9%        ...
1991   12%        ...
         ⋮
1999    7%        ...
```

⇒

```
<header>Australian unemployment</header>
<date><header>Year</header> 1990 1991 ...  1999</date>
<number><header>Level</header> 0.09 0.12 ...  0.07</number>
<string><header>Notes</header> ...  </string>
```

⟨ unemployment rate Australia 1990s ⟩ ⇒

```
#or(unemployment.string unemployment.header)
#or(rate.string rate.header)
#or(australia*.placename australia*.header)
#or(1990*.date 1991*.date 1992*.date 1993*.date 1994*.date 1995*.date
1996*.date 1997*.date 1998*.date 1999*.date)
```

Figure 2: The results of type inference and rewriting for Table 1a, and for our example query.

Their approach is similar to ours, in that table rows and captions are marked up prior to indexing; but there is no rewriting, the unit of indexing is the row (or cell in later work), and the usual full-text index mechanisms are used. Some type identification is used in the QuASM system [4], in both tables and queries, but this was dropped in later work. Their evaluations demonstrate a substantial improvement in question-answering performance: however, table extraction and ranking are conflated and results "should be viewed more as a test of extraction".

The "TableSeer" method [3] also attempts to rank tables directly in response to a query, but does so on the basis of a full-text index of extracted metadata (captions and headers) without the extended matching described here. The interface allows queries over caption, column headings, content (as text), as well as the number of rows or columns in the extracted table, and ranking is on the basis of a tf.idf variant plus priors on each table and source document. Accuracy was reported as high as 70%, over a set of 20 "random" queries, although the definition of "accuracy" is not clear.

5. SUMMARY AND FURTHER WORK

Despite the growing importance of online data repositories, and tabular data in particular, present portals provide very poor search capabilities. A naïve approach, simply treating a table as a text document and applying the usual techniques of ad-hoc retrieval, is not much improvement and suffers from both false positives and false negatives.

Our first attempt at a ranking algorithm for tables uses simple type inference and keeps columns of data together, and rewrites both tables and queries to a simple alternative form. This improves early success by 6–8 points on a simple testbed.

Much further work is possible. There are some common table features which our parser cannot handle; in particular, header rows for sections within the table, certain types of cross-tabulation, and files which include more than one table. Each of these would require further development in the parser. Further, we do not consider the problem of extracting tables from free text. Adaptations of existing, smarter, algorithms [e.g. 1, 3, 7] would help here.

Additional data types will be required to extend the parser's use, including for example gender, postcodes, and date ranges. We have also not yet considered potential ranking factors such as duplication, table size, table coverage, and recency—indeed it is not at all clear what the "right" behaviours are here.

There is also a clear need for a larger and more representative test set. We plan to build such a set, and hope to acquire query logs from several portals to inform the process. A more comprehensive evaluation would also inevitably suggest further avenues for improvement in the parser, query re-writer, and ranking.

	With table headers			No table headers		
	S@3	S@5	MRR	S@3	S@5	MRR
Full-text	.67	.77	.69	.53	.62	.56
Typed columns	.67	.85	.70	.60	.68	.60

Table 2: Effectiveness metrics on our preliminary testbed, for a full-text indexing approach and for the approach described here.

Finally, finding the right tables is at best half the problem. There is no version of query-biased summaries for tables—portals tend to show the metadata alone. However, since our strategy indicates which columns of which tables best match the query, we can in principle provide query-biased summaries either through natural language generation, small summary tables, or graphical overviews. This is the subject of current research.

Acknowledgment. We thank those colleagues who provided sample queries.

References

[1] Marco D Adelfio and Hanan Samet. Schema extraction for tabular data on the web. *Proc. VLDB Endowment*, 6(6):421–432, 2013.

[2] David W Embley, Matthew Hurst, Daniel Lopresti, and George Nagy. Table-processing paradigms: a research survey. *Int. J of Document Analysis and Recognition*, 8(2–3):66–86, 2006.

[3] Ying Liu, Kun Bai, Prasenjit Mitra, and C. Lee Giles. TableSeer: Automatic table metadata extraction and searching in digital libraries. In *Proc. JCDL*, pages 91–100, 2007.

[4] David Pinto, Michael Branstein, Ryan Coleman, W. Bruce Croft, Matthew King, Wei Li, and Xing Wei. QuASM: A system for question answering using semi-structured data. In *Proc. JCDL*, pages 46–55, 2002.

[5] Pallavi Pyreddy and W Bruce Croft. TINTIN: A system for retrieval in text tables. In *Proc. ACM Int. Conf. on Digital Libraries*, pages 193–200, 1997.

[6] Trevor Strohman, Donald Metzler, Howard Turtle, and W Bruce Croft. Indri: A language model-based search engine for complex queries. In *Proc. Int. Conf. on Intelligent Analysis*, volume 2, pages 2–6, 2005.

[7] Xing Wei, Bruce Croft, and Andrew McCallum. Table extraction for answer retrieval. *Inf. Retrieval*, 9:589–611, 2006.