FACTO: A Fact Lookup Engine Based on Web Tables

Xiaoxin Yin, Wenzhao Tan, Chao Liu
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{xyin, wentan, chaoliu}@microsoft.com

ABSTRACT

Recently answers for fact lookup queries have appeared on major search engines. For example, for the query {Barack Obama date of birth} Google directly shows "4 August 1961" above its regular results. In this paper, we describe FACTO, an end-to-end system for answering fact lookup queries for web search. FACTO extracts structured data from tables on the web, aggregates and cleans such data and stores them in a database. Given a web search query, FACTO will decide if it asks for facts in this database, and provides the most confident answer when possible. FACTO achieves higher precision and comparable coverage comparing with the fact lookup engines by Google and Ask.com, although FACTO is developed by a very small team. We present the challenges and our solutions in developing every component of FACTO. Some solutions are based on existing technologies, and many others are novel approaches proposed by us.

Categories and Subject Descriptors

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

General Terms

Algorithms, Measurement, Experimentation.

Keywords

Web search, Question answering, Information extraction.

1. INTRODUCTION

Recently we have witnessed more and more usage of structured data by search engines, and a most visible feature is the *direct answers to fact lookup queries*. For example, for queries {Barack Obama date of birth}, {morocco capital} and {ps3 release date} (we use "{}" to represent a query), Google returns direct answers above the regular search results (as in Figure 1). Such direct answers appear for queries looking for many kinds of facts, including directors of movies, spouses of celebrities, and population of cities. Such answers bring great convenience to users searching for facts on the web, because it is no longer necessary for the user to inspect the snippets, click into a page and find the answer in the text. It will not be surprising if direct answers can change users' search habits from searching for web pages to searching for facts. Although there is clear user need, the fact lookup direct answers do not appear on major search engines until recently, which indicates that building a mature fact lookup engine is a challenging task. Today the precision and coverage of the fact lookup answers on major search engines are still quite limited. For example, the third answer in Figure 1 is incorrect as PlayStation 3 has been on the market for four years. On the other hand, Google does not show direct answers for many fact lookup queries. For example, it does not show such an answer for {britney spears birth date}, although it shows an answer for {britney spears date of birth}.

Figure 1: Three examples of Google's direct answers. The first two answers are correct and the third is wrong.

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0632-4/11/03.

In this paper we introduce FACTO, a fact lookup engine based on structured data from the web. FACTO is designed to work within the search engine, and it should only show answers for queries looking for facts. Comparing with the fact lookup engines of Google and Ask.com, FACTO achieves higher precision and comparable query coverage (higher than Google and lower than Ask.com), although it is built by a very small team of two people in less than a year. According to the best of our knowledge, this is the first paper that describes an end-to-end system for answering fact lookup queries in search engines.

Answering natural language questions has been studied in information retrieval and natural language processing [3][11][17][18]. However, this is very different from identifying and answering fact lookup queries in web search. Web users are lazy and seldom submit natural language queries. More importantly, these question answering approaches cannot distinguish fact lookup queries from other queries, and they usually treat every query as a question. In web search only a small portion of queries are fact lookup queries. If we assume every query is looking for facts, very likely we will provide irrelevant answers to most of them.

Unlike the above question answering approaches, FACTO relies on structured data extracted from the web. The main philosophy of FACTO is to extract entity-attribute-value triples from tables on the web, and use such data to answer queries that ask for an attribute of an entity. FACTO extracts data from tables containing attribute-value pairs, such as the table in Figure 2 (a). We choose to use attribute-value tables because such tables widely exist on the web, and the number is much larger than that of two-dimensional tables (such as the table in Figure 2 (b)). Attribute-value tables provide very rich data covering all kinds of domains, which enables us to build a generic fact lookup engine. However, it is not sufficient to simply extract attribute-value pairs from tables, because the entity involved is seldom in the same table. We propose a novel approach to infer the main entity of a web page, by matching web search queries with the titles and contents of the search result pages clicked for these queries. By combining

Type	Public (NASDAQ: COST)
Industry	Retailing (Warehouse club)
Founded	1983
Founder(s)	James D. Sinegal Jeffrey H. Brotman
Headquarters	Issaquah, Washington, U.S.
Number of locations	563 (2009)
Area served	Worldwide
Key people	James Sinegal, Founder & CEO Jeffrey Brotman, Founder & Chairman W. Craig Jelinek, President/COO

(a) An attribute-value table on Wikipedia page of Costco

Player	TEAM	POS	G	AB	R	H
1. I Suzuki	SEA	OF	162	680	74	214
2. C Figgins	SEA	2B	161	602	62	156
3. J Lopez	SEA	3B	150	593	49	142
4. F Gutierrez	SEA	OF	152	568	61	139
5. C Kotchman	SEA	1B	125	414	37	90
6. R Branyan	SEA	DH	109	376	47	89
7. J Wilson	SEA	SS	108	361	22	82
8. J Smoak	SEA	1B	100	348	40	76
9. M Saunders	SEA	OF	100	289	29	61
10. M Bradley	SEA	OF	73	244	28	50
11. A Moore	SEA	C	60	205	12	40
12. J Wilson	SEA	SS	61	193	17	48
13. R Johnson	SEA	C	61	178	24	34
14. M Tuiasosopo	SEA	OF	50	127	12	22
15. J Bard	SEA	C	39	112	9	24
16. R Langerhans	SEA	OF	60	107	16	21
17. M Sweeney	SEA	DH	30	99	11	26
18. K Griffey	SEA	DH	33	98	6	18
19. E Alfonzo	SEA	C	13	41	4	9
20. M Mangini	SEA	3B	11	38	2	8
21. M Carp	SEA	1B	14	37	1	7
22. E Byrnes	SEA	OF	15	32	1	3
23. G Halman	SEA	OF	9	29	1	4
24. C Woodward	SEA	SS	8	19	0	3
25. G Quiroz	SEA	C	2	7	1	2

(b) A two-dimensional table on seattle.mariners.mlb.com

Figure 2: An example of attribute-value table and that of two-dimensional table

entities with attribute-value pairs, we create a very large repository of entity-attribute-value triples, which is a most convenient way to store facts in a domain-independent way[1]. There are significant challenges in almost every step of building this data repository, such as how to identify attribute-value tables, how to find main entity of each page, and how to remove noise from the data.

With the repository of entity-attribute-value triples, FACTO can provide direct answers to queries consisted of an entity and an attribute, such as {Madonna's birth date}. The major challenges in query answering include how to answer queries in which users use an entity name or attribute name that is different from our data, how to judge if a query really contains an entity-attribute pair or just happens to be consisted of an entity name and an attribute name, and how to select the value with highest confidence.

Here is a list of major contributions we made in this paper.

- We build FACTO, an end-to-end system for answering fact lookup queries in web search, which achieves higher precision and comparable query coverage comparing with fact lookup engines of Google and Ask.com. We evaluate the performance of each of the many components of FACTO, as well as that of the whole system.
- We propose a new method to extract entity-attribute-value triples from web pages. Although attribute-value tables can be identified using existing methods, such data is not useful unless we find the corresponding entities, which seldom appear in the tables. We find the important entities in web pages by analyzing the web search queries and the pages clicked for them. We can also extract entities from pages with few or no clicks utilizing the fact that many pages from the same web site have identical format. Our experiments show FACTO achieves high accuracy in extracting data from the web.
- A major challenge in answering fact lookup queries is how to judge if a query is really looking for some fact about an entity, or happens to have an entity name and an attribute name. We apply Authority-Hub analysis [13] by treating entity-attribute pairs as authorities and data sources as hubs. An entity-attribute pair provided by many data sources will have high authority score, and a data source providing many authoritative entity-attribute pairs will have high hub score. Only the entity-attribute pairs with high authority scores are considered by FACTO.

[1] Freebase stores its data in a similar way, as shown in http://wiki.freebase.com/wiki/Data_dumps.

Figure 3: User Interface of FACTO

We conduct an empirical study for each new approach proposed in this paper. We also compare the precision and query coverage of FACTO, Google, and Ask.com, and observe that FACTO achieves higher precision and comparable query coverage. A snapshot of FACTO is shown in Figure 3. FACTO can be accessed at http://lepton.research.microsoft.com/facto/, with examples at http://lepton.research.microsoft.com/facto/about.htm.

The remaining of this paper is organized as follows. Related work is discussed in Section 2. In Section 3 we present our approaches for extracting attribute-values and entities from the web. Section 4 describes the query answering component, including how to find equivalent names of entities and attributes. Experimental results are presented along with the technologies in different sections. We conclude this study in Section 5.

2. RELATED WORK

Answering fact lookup questions has been widely studied in information retrieval and natural language processing, with a good survey in [17]. Studies in this area focus on natural language questions, such as "When was James Dean born" and "Who founded the Black Panthers organization". Different approaches have been proposed, including those based on knowledge base and ontology [11], those based on document retrieval plus answer extraction [18], and those relying on search engines [3]. However, answering natural language questions is very different from identifying and answering fact lookup queries in web search. First, web users are lazy and seldom submit natural language queries, – they tend to send queries like {morocco capital} instead of {where is the capital of morocco}. It is difficult to infer whether a question is being asked and what is being asked from such a short query. Second and more importantly, the above question answering approaches cannot distinguish fact lookup queries from other queries and assume all queries are fact lookup queries. But in web search only a small portion of queries are fact lookup queries, while the majorities are navigational queries, entity queries, transactional queries, etc. If we assume every query is a fact lookup query, very likely we will provide irrelevant answers to most of them.

Web table analysis has been widely studied. Many studies focus on how to distinguish "genuine" tables from the tables for HTML layout purposes. Wang and Hu [20] propose a machine learning based approach for genuine table classification, which uses features such as mean and standard deviation of numbers of rows and columns. Similar features are used in the approach by Cafarella et al. [4], which discovers tables together with their column headers, so that tables on the web can be converted into tables in a rela-

tional database. Gatterbauer et al. [10] use visual features of HTML tables in classification. Tengli et al. [19] propose approaches for distinguishing attribute cells and value cells in tables, in order to extract tabular data with semantics. Crestan and Pantel study how to classify attribute-value tables in their very recent work [7]. However, they do not have an accurate way to find the entity that is the subject of each attribute-value table. We solve this problem by matching queries with clicked pages and utilizing sets of pages with same format, which achieves high accuracy.

There are also studies on extracting information from very large document collections. SnowBall [1] starts from a small set of seed facts and keeps expanding it by learning natural language patterns and applying them on text. Paşca et al. propose an approach with the same philosophy [16], and use it to extract one million facts of birth year of people using a seed set of ten facts. These facts are sorted by their confidences and their accuracy ranges from 98.5% (facts with highest confidences) to 75% (facts with lowest confidences). KNOWITALL [9] expands the set of facts using information on the web, by submitting appropriate queries to search engines. TextRunner [2][21] extracts 7.8M facts which are assertions like (*Edision, invented, light bulbs*) from 9M web pages, with accuracy 80.4%. WEBTABLES [5] extracts 154M data tables from billions of documents crawled by Google, and allows users to search these tables and find related attributes or tables.

FACTO shares some common philosophy with some approaches mentioned above, such as using machine learning to distinguish tables containing attribute-values, and exploring attribute correlations which is also studied in [5]. But in general FACTO is very different as it is an end-to-end fact lookup engine, which starts from data extraction from web and can answer fact lookup queries with high precision.

3. DATA EXTRACTION

FACTO contains two major components: Data extractor that extracts entity-attribute-value triples from the web, and query answering engine which identifies the queries looking for facts about entities and answers with the data extracted. In this section we will describe the data extractor.

The architecture of the data extractor is shown in Figure 4. The first component is the *table classifier* which distinguishes attribute-value tables from the vast majority of other tables. The problems of identifying "genuine" data tables has been widely studied in [4][7][10][19][20], and we adopt a machine learning based approach similar to that in [4], as described in Section 3.2.

The second component is the *URL pattern summarizer*, which finds sets of pages with same format [22]. Many web sites contain large sets of pages with same format. We treat each set of pages of same formats as a *data source*. By analyzing data from different pages in a data source, we can filter out many false positive attribute-value tables, as described in Section 3.5.

The third component is the *entity extractor* that is described in Section 3.4. It extracts the main entity of each web page by analyzing web search queries with clicks on this page or pages of same format. For a page that is clicked as a web search result, its main entity can often be found by matching user queries with the page title, header, or contents. For example, a user may search for {Costco company} and click http://en.wikipedia.org/wiki/Costco (the page containing the table in Figure 2 (a)), and we can infer the main entity for this page is "Costco", which appears in both the query and the page title. For pages that receive few or no clicks, we identify their main entities by analyzing pages with the same format and learning how to extract entity names by building HTML wrappers.

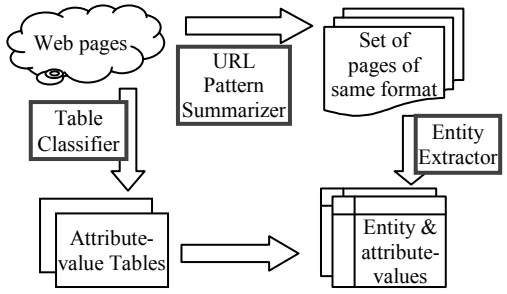


Figure 4: Overview of Data Extractor

The final step is to join the entities with the attribute-values extracted from tables, in order to create a large repository of entity-attribute-value triples.

3.1 A Survey of Tables on the Web

Before describing our approach, we first explain why we choose attribute-value tables as our main data source. According to our observation, a significant portion of tables on the web are attribute-value tables, with first column containing attribute names and second column containing values. Cafarella et al. [5] have shown the web contains a huge number of relational tables, i.e., two-dimensional tables providing values of multiple entities on multiple attributes. We perform an empirical study to compare the amount of both types of tables. We randomly select 1000 HTML tables from all indexed pages of Bing, and manually identify each type of tables from them. The statistics of each type of tables are shown in Table 1. The number of attribute-value tables is about 4 times that of relational tables (6.6% vs. 1.6%). This is fairly consistent with the number of attribute-value tables (744M) extracted by our table classifier and the number of relational tables reported by Cafarella et al. in their 2008 paper [5]. On the other hand, each relational table contains about twice more data elements. We also find 63% of data in relational tables are numerical, while this ratio is only 9% in attribute-value tables.

In general, there is a very large amount of data in attribute-value tables, and the data amount is similar to that of two-dimensional tables. These two types of tables contain different types of data, – there are more textual data in attribute-value tables and more numerical data in two-dimensional tables. We use attribute-value tables as our data source because most fact lookup queries look for non-numerical facts, such as people or locations.

Table 1: Comparing attribute-value tables and relational tables

	Attr-value tables	**Relational tables**
Total number	744M (from 20B web pages)	154M (reported by Cafarella et al. [5])
Percentage (in randomly selected tables)	6.6%	1.6%
Avg. # entities	1	10.31
Avg. # attributes	14.38	3.63
Avg. # data elements	14.38	38.88
Percentage of numerical data elements	8.85%	62.86%

3.2 Table Classification

The problem of how to distinguish data tables and non-data tables has been studied in [4][7][10][20], and we just adopt the approach

in [4] with a different feature set, which is designed for distinguishing attribute-value tables. The feature set includes features such as numbers of non-empty cells in the first and second columns, mean and standard deviation of number of non-empty cells in each row, those of text length of cells in each column, and percentage of non-empty cells and distinct cells in each column.

We use SVM to train classifiers for distinguishing attribute-value tables, based on a manually labeled training set containing 138 positive examples (attribute-value tables) and 523 negative ones. libSVM [6] is used with four types of kernels: Linear, Polynomial, RBF and Sigmoid. We use the 1000 randomly selected HTML tables to test the accuracy of our table classifier. They are manually labeled and 66 of them are attribute-value tables. SVM with a linear kernel achieves highest F1 of 0.759 (precision 0.696 and recall 0.833), and thus we use it in FACTO.

3.3 Summarizing URL Patterns

It is very common for a web site to contain many pages in the same format, which contain same or similar sets of attributes for different entities. For example, celebrina.com has a page for each company (e.g., http://www.celebrina.com/madonna.html for Madonna). These pages have the same format and each of them contains an attribute-value table with the same set of attributes. Such sets of uniformly formatted pages also exist on many popular web sites such as Wikipedia, facebook and dpreview. They are very useful in data extraction, as we can learn the format from some pages and use that to extract data from other pages.

We use an approach from our previous work [22] for finding pages of the same format, and we briefly describe it here for completeness. In [22] it is found that web pages from a domain can be automatically divided into many groups simply based on their URLs. This approach has high accuracy in finding sets of pages in the same format, – experiments show that in about 96% of cases two URLs in the same group have same format. This approach is based on *URL patterns*, which are regular expressions that can match with URL strings. For example, http://www.imdb.com/name/nm*/bio is a URL pattern, which matches with all IMDB URLs of people's biographies. Because each URL pattern usually contains a set of pages containing same or similar sets of attributes for different entities, we treat each URL pattern as a *data source*. Many of the following analyses will be applied on data from each data source separately.

3.4 Extracting Entities

The attribute-values extracted from a web page are not useful unless we can identify which entity they are about, which usually needs to be extracted from somewhere else in the page. According to our observation and experiments (see Section 3.6), the attribute-values from a page are usually about the entity that this page talks about, and we call such entity as the *main entity of the page*. It is a challenging task to identify the main entity of a page, as a page may contain many entities.

For each web page, usually there is only one entity that the web search users query for, which is the main entity of the page. Therefore, the main entity can often be found by matching user queries with clicks on this page with the page contents. For example, a user may search for "Britney Spears music" and click on http://www.last.fm/music/Britney+Spears, whose title is "Britney Spears – Discover music, videos, concerts, & pictures at Last.fm". The longest common substring of the query and the title is "Britney Spears", which is the entity that this page is about. More importantly, every artist's page on www.last.fm (i.e., every page following URL pattern http://www.last.fm/music/*) has a title like "(artist's name) – Discover music, videos, concerts, & pictures at Last.fm". If we can find this from a few such pages based on user queries, we can extract the main entity from every such page.

FACTO has two components for entity extraction: Wrapper builder and wrapper ranker. The wrapper builder is responsible for finding the main entity of a page given a user query with click(s) on that page, and building a wrapper for it. It tries to match the query with the page title and the text in each <h1> element. If no match is found, it tries to match with each <h2>, and so on, until it finds a match or has tried each element in the page. When matching a piece of text (e.g., title of a page) with a query, we look for the longest sequence of words in the text, so that each word appears in the query except stop words (e.g., "a", "the", "is", "for"). This longest sequence is treated as the entity name, and a regular expression is built by replacing this entity name with "(*)", which indicates the part to be extracted.

FACTO uses wrappers based on HTML tag-paths [15]. For example, when matching query {Britney Spears music} with her page on Last.fm, FACTO builds two wrappers, one from title and one from header. The one from title is "<html><head><title>(*) – Discover music, videos, concerts, & pictures at Last.fm", and the one from header is "<html><body> <h1>(*)". Both can extract "Britney Spears" as the entity name.

The second component for entity extraction is wrapper ranker. Usually many wrappers are built for web pages from a data source. Some of them are incorrect for different reasons, such as some user queries not containing the complete entity name. We need to rank these wrappers and choose the entity extracted from the best wrapper applicable to each page.

In the ideal case wrappers should be ranked by precision and recall. But we cannot directly measure them because we do not use labeled data. We try to measure precision and recall based on the pages from which entities can be extracted using user queries. For each web page d and query q with clicks on d, let $\text{match}(d, q)$ be the entity found by matching q with d. Let $Q(d)$ be the set of distinct queries with clicks on d. For any entity e extracted from d, we estimate the probability of e being correct as

$$p^+(e) = \frac{|\{q|q \in Q(d), \text{match}(d,q) = e\}|}{|Q(d)|}, \quad (1)$$

which is the portion of queries in $Q(d)$ from which e can be inferred as the entity. Let D be the set of web pages of a data source. Let w be a wrapper for D and $w(d)$ be the entity extracted by applying w on d. The precision of w can be calculated by

$$\text{precision}(w) = \frac{\sum_{d \in D, w(d) \neq \emptyset} p^+(w(d))}{|\{d|d \in D, w(d) \neq \emptyset\}|}, \quad (2)$$

The recall of w can be calculated by

$$\text{recall}(w) = \frac{\sum_{d \in D, w(d) \neq \emptyset} p^+(w(d))}{|\{d|d \in D, \exists w', w'(d) \neq \emptyset\}|}, \quad (3)$$

The score of wrapper w is defined as the harmonic mean of precision and recall, i.e.,

$$\text{score}(w) = \frac{2 \cdot \text{precision}(w) \cdot \text{recall}(w)}{\text{precision}(w) + \text{recall}(w)}. \quad (4)$$

For each web page, we extract the main entity using the wrapper with highest score.

3.5 Extracting Attribute-Values

The last step of data extraction is to extract attribute-value pairs from attribute-value tables, and combine them with the main entity from each page. It is rather straight-forward to extract attributes and values from the first and second column of an attribute-value table. However, our table classifier only considers the contents in

an individual table, which is often insufficient to make the right decision. Some tables classified as attribute-value tables are false positives. We find the majority of these false positives fall into two categories. The first category contains tables with same contents that appear on many pages, such as the table in Figure 5 (a). The second category contains tables containing a set of values without attributes, such as the table in Figure 5 (b).

Log in	Contact us
Help	Customer services
About us	Store locations

(a)

Britney Spears	Paris Hilton
Jennifer Lopez	Jessica Simpson
Madonna	Jessica Alba

(b)

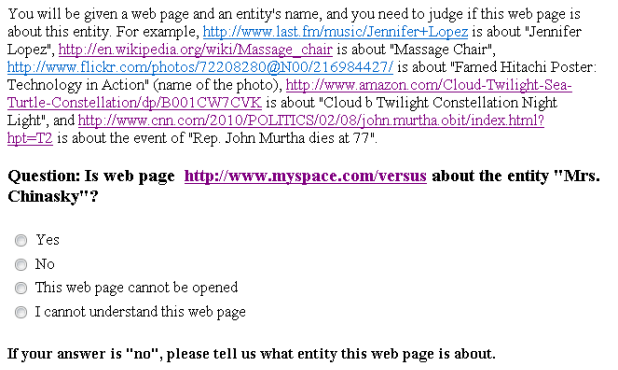
Figure 5: Examples of false positives of attribute-value tables

Both types of false positives can be detected by computing statistics of attribute-value pairs extracted from pages in a data source. For each attribute appearing in a data source, we compute (1) entropy of all values of this attribute, and (2) number of pages containing this attribute. If the values of an attribute have very low entropy, this attribute contains little information and is very likely to belong to the first category. If an attribute appears in a small number of pages, it is unlikely to be a real attribute, which should exist in multiple entities. We ignore an attribute if it appears in less than five pages or its values have entropy less than 0.5. A table is ignored if it contains no useful attributes.

3.6 Experiments on Entity and Attribute-Value Extraction

Here we report the experiment results on extracting entity and attribute-values. We extract 744M attribute-value tables from 20B web pages in Bing's index on 2010/06/22. These tables come from 417M web pages. To extract entities we use all query-click logs from U.S. market during 2008/08/01 to 2009/05/31, which contains each search query and all URLs clicked for it by any user. Among the 417M web pages containing attribute-value tables, we extract an entity from each of 93.3M pages, which contain 164M attribute-value tables and 749M attribute-value pairs. No entity is extracted from majority of pages because majority of pages are not in English and thus have no or very few clicks in the logs of U.S. market. After removing noise according to Section 3.5, there are 603M entity-attribute-value triples left.

We first test the accuracy of FACTO in entity extraction. We manually inspect 50 uniformly randomly selected entities, and use Amazon Mechanical Turk (MTurk) to inspect 500 of them, in order to see if they are the main entities in the corresponding web pages. An example MTurk question is shown in Figure 6. Each question is answered by three workers of MTurk, and all answers of "cannot access page" are ignored unless all three workers say so. The accuracy according to voted result for each entity is shown in Table 2, and the average accuracy is 97.4%.


You will be given a web page and an entity's name, and you need to judge if this web page is about this entity. For example, http://www.last.fm/music/Jennifer+Lopez is about "Jennifer Lopez", http://en.wikipedia.org/wiki/Massage_chair is about "Massage Chair", http://www.flickr.com/photos/72208280@N00/216984427/ is about "Famed Hitachi Poster: Technology in Action" (name of the photo), http://www.amazon.com/Cloud-Twilight-Sea-Turtle-Constellation/dp/B001CW7CVK is about "Cloud b Twilight Constellation Night Light", and http://www.cnn.com/2010/POLITICS/02/08/john.murtha.obit/index.html?hpt=T2 is about the event of "Rep. John Murtha dies at 77".

Question: Is web page http://www.myspace.com/versus about the entity "Mrs. Chinasky"?

- Yes
- No
- This web page cannot be opened
- I cannot understand this web page

If your answer is "no", please tell us what entity this web page is about.

Figure 6: A question for entity relevance on Mechanical Turk

Table 2: Accuracy of Entity Extraction

	Relevant	Irrelevant	Tie	Cannot access page
MTurk	486	12	0	2
Manual	48	2	0	0
Excluding cases of "tie" and "cannot access page"				
#cases	Avg. accuracy	95% conf. interval of accuracy		
548	97.4%	[96.1%, 98.8%]		

We also test whether the extracted tables contain attribute-values relevant to the specified entity. Again we manually inspect 50 uniformly randomly selected tables, and use MTurk to inspect 500 of them. For each table, we ask the worker to judge if it contains attribute-values of the specified entity. The results are shown in Table 3, and average accuracy is 89.2%.

Table 3: Accuracy of Attribute-Value Tables

	Relevant	Irrelevant	Tie	No such page, table, or entity
MTurk	421	52	26	1
Manual	43	4	0	3
Excluding cases of "tie" and "No such page, table, or entity"				
#cases	Avg. accuracy	95% conf. interval of accuracy		
520	89.2%	[86.6%, 91.9%]		

Here we show some typical attribute-value tables of two randomly chosen entities. The first entity is "Ameritrade", which has ten tables extracted, and here are two of them:

http://www.linkedin.com/companies/ameritrade?lnk=vw_cprofile

Headquarters	Greater Omaha Area
Industry	Financial Services
Type	Public Company
Status	Operating Subsidiary
Company Size	1001-5000 employees
Founded	1997
Website	http://www.ameritrade.com

http://www.wikinvest.com/wiki/Ameritrade

P/E ►	16.4	AVG
EV/EBITDA ►	10.4	AVG
ROA ►	4.10%	AVG
ROE ►	21.50%	HIGH
Debt to Equity ►	4.1	AVG
Interest Coverage Ratio ►	12.9	HIGH

The second entity is "Atlantic City", with 156 attribute-value tables extracted, and here are two examples.

http://eppraisal.com/localpages/new%20jersey/atlantic%20city.aspx

Average Temp in January:	31.7 degrees
Average Temp in July:	76 degrees
Average Snowfall:	16.3 inches
Average Precipitation:	45.11 inches
Average Sunny Days:	56

http://fr.weather.yahoo.com/usnj/usnj0015/index_c.html

Humidité:	70%
Vent:	N/13 km/h
Visibilité:	16.09 km
Point de rosée:	11°
Pression:	1019 mb
Lever du soleil :	6:49
Coucher du soleil	18:50

4. QUERY ANSWERING

After extracting entity-attribute-value data from the web, we store them in a relational database with an index built on entity and attribute. Given a web search query, FACTO tries every way to interpret it as a fact lookup query by considering part of the query as an entity and part as an attribute. A major challenge is how to judge if a query is really a fact lookup query, or happens to contain an entity name and an attribute name. For example, FACTO extracts an entity "Red" with an attribute "hair" from http://dcanimated.wikia.com/wiki/Red, and thus it will provide an answer for query {red hair}, which is not a fact lookup query. We

use Authority-Hub analysis [13] by treating entity-attribute pairs as authorities and data sources as hubs. An entity-attribute pair provided by many data sources will have high authority score, and a data source providing many authoritative entity-attribute pairs will have high hub score. We select the entity-attribute pairs with high authority scores as the ones to be answered by FACTO.

Another challenge is that users often use entity/attribute names that are different from those in our data. For example, our database contains "date of birth" of "Britney Spears", and users may query with {britney date of birth} or {dob of britney spears}. We use an approach for detecting similar queries [8] to detect equivalent entity names, because entity names are often used as queries. In contrast, attribute names are not directly used as queries, and we detect equivalent attribute names based on the observation that an entity should have same or similar values on the same attribute in two data sources. If two attributes are often associated with the same value for same entity, they are likely to be equivalent.

For each possible interpretation, FACTO queries the database using the entity and attribute and also their equivalent forms. If certain combinations of entity and attribute exist in the database, FACTO retrieves the corresponding values and selects a value with highest confidence among all retrieved values, which is returned as the answer of the query. This procedure is illustrated in Figure 7. We will describe each component in this section.

4.1 Query Interpretation

We use a rule-based approach for interpreting queries. There exist more advanced approaches on query parsing such as [14]. But we choose a simple, rule-based approach because approaches like [14] require manually labeled training data that is not available to us. Moreover, web users tend to use short and simple queries instead, and a more advanced query parsing approach is unlikely to improve the query coverage significantly.

We use *e* to denote an entity and *a* an attribute. The following five rules are used in FACTO to convert a query into an entity-attribute pair: "*e a*", "*e*'s *a*", "(the)? *a* of *e*", "(what|who|when|where) (is|are|was|were) (the)? *a* of (the)? *e*", "(what|who|when|where) (is|are|was|were) *e*'s *a*". We also have four specific rules for "how long", "how old", "when was someone born" and "where was someone born".

Given a user query, FACTO finds all possible interpretations of entity-attribute pairs by matching it with every rule in every possible way. FACTO also maintains the list of all attributes in main memory, so that any interpretation without an existing attribute is discarded. All remaining interpretations will go through the following procedure for query answering.

4.2 Entity-Attribute Filtering

There are many queries that happen to contain an entity name and an attribute name. For example, from http://en.wikipedia.org/wiki/Sand_(film) FACTO extracts an entity "Sand" with an attribute "language", and it may provide an answer for query {sand language}, which actually looks for a book called "The Language of Sand". We say an entity-attribute pair is *valid* if (1) this attribute exists for this entity, and (2) the query constructed by concatenating the entity and the attribute asks for the specific attribute of the entity. For example, ("China", "capital") is a valid entity-attribute pair, while ("Sand", "language") and ("Seattle", "WA") are not. When the intent of a query is not obvious, we infer it from the clicked URLs of Bing.

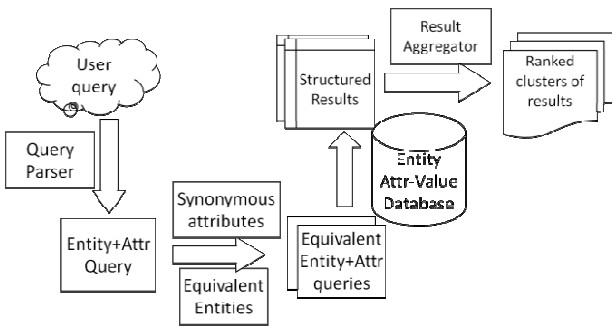


Figure 7: Overview of Query Answering System

We observe that if an entity-attribute pair is provided by many data sources, it is more likely to be a valid pair. This problem is analogous to Authority-Hub analysis [13] by considering entity-attribute pairs as authorities and data sources as hubs. An entity-attribute pair provided by many data sources is likely to be valid and popular, and should have high authority score. Similarly, a data source providing many authoritative entity-attribute pairs should have high hub score. We implement the iterative method for computing authority-hub scores using MapReduce on a distributed environment of Microsoft [12]. Initially each data source has score of 1. In each iteration we compute the scores of entity-attribute pairs and then those of the data sources. After each iteration we compute the relative change of scores, which is defined based on the vector of scores of all entity-attribute pairs. We keep iterating until the relative change drops below 0.001, which requires five iterations in this problem.

We test the accuracy of this approach by studying if it assigns higher score to valid entity-attribute pairs than to invalid ones. We randomly select 25 entities with at least 10 attributes, and select a valid and an invalid attribute for each entity. In this way we get 25 valid entity-attribute pairs and 25 invalid ones. For each valid pair and each invalid pair (either for same entity or not), we test if our approach assigns higher score to the valid pair. The accuracy is defined as the percentage of cases our approach is correct. The accuracy is 85.74%.

There are 509M entity-attribute pairs. After inspecting the data, we use the 73M pairs with highest authority scores as the entity-attribute pairs that FACTO will provide answers for.

4.3 Attribute Equivalence

One major challenge in using extracted data to answer queries is that users may use an attribute name that is different from our data. Similarly, different web sites often use different names for the same attribute, such as "birth date" and "date of birth", "director" and "directed by". In this section we study how to identify equivalent attributes from different web sites, which will help us match the attribute names in user queries with those in our data.

Our key observation is that if an entity appears in multiple data sources, it should have same or similar values on two equivalent attributes. Therefore, if two attributes from two data sources usually have same value on same entity, they are likely to be equivalent. However, we do not know if two entities in different data sources are the same, even if they have same name. Thus we also need to infer equivalence between entities based on their attributes.

This problem has been studied in [23], which aims at finding equivalent attribute names across two databases. It infers the similarities between attributes according to whether they have same value on same entity, and infers the similarities between entities based on whether they have same value on same attribute. This

Table 4: Accuracy of Equivalent Attribute Detection

Two attributes are	*Count*	*%Correct*	*Example of two attributes*
Any	50	88%	
Both in English	26	88.5%	"population" vs. "population (estimate)" of "zephyrhills"
One in English	14	92.9%	"family" vs. "familie" (Dutch) of "paeonia"
Neither in English	10	80%	"время ожидания" (waiting) vs. "в режиме ожидания" (standby) of "nokia 6260 slide"

problem is then converted into Authority-Hub analysis [13] by treating attribute-pairs as authorities and entity-pairs as hubs.

We implement a variant of this approach using MapReduce and apply it to the entity-attribute-value triples extracted from the Web. For each pair of attributes (or each pair of entities), we compute their similarity based on their values on different entities (or attributes). An iterative procedure is used, which assigns initial similarity of 1 to all pairs of entities with same name, and keeps computing the similarities between attributes and those between entities based on each other. We ignore pairs of entities with different names because (1) there are too many entities and it is impossible to compute for each pair of them, and (2) our goal is to detect equivalence between attributes.

The above approach can detect equivalent attributes in different data sources. To answer queries we need to know equivalent attribute names independent of any data source. For example, we need to know "dob" and "date of birth" are equivalent, instead of "dob" from a data source and "date of birth" from another. This is done by aggregating the equivalence relationships between attributes from different data sources.

We define the similarity between two attribute names a_1 and a_2 as the weighted average of similarities between pairs of attributes with these two names in different data sources:

$$\mathrm{sim}(a_1 = a_2) = \frac{\sum_{n(a_{ig})=a_1, n(a_{kh})=a_2} w(a_{ig}, a_{kh}) \cdot P(a_{ig}=a_{kh})}{\sum_{n(a_{ig})=a_1, n(a_{kh})=a_2} w(a_{ig}, a_{kh})} \quad (7)$$

, where $n(a_{ig})$ is the name of attribute a_{ig} and $w(a_{ig}, a_{kh}) = \log(|\{(e_1, e_2) | e_1.a_{ig} = e_2.a_{kh}\}|)$.

Here we evaluate the equivalent attributes detected. The total number of attributes in different data sources is 1.15M. FACTO analyzes the 2.99M pairs of attributes (each in a different data source), with each pair having the same value on at least one entity. It finds 1.26M pairs of attributes with similarity above 0.5.

We evaluate the accuracy of equivalent attributes by uniformly randomly selecting 50 pairs of attributes and manually label if each pair are truly equivalent. Each pair of attributes come from different data sources, and in 6 of the 50 pairs the attributes have same name. We do not use MTurk because many attribute names are not in English, and MTurk workers may not be diligent enough to find out their meanings. Google Language Detection (http://www.google.com/uds/samples/language/detect.html) is used to detect the language of a non-English attribute name.

Table 5: Synonymous attribute names

address	*phone*	*price*	*weight*
location	telephone	list price	gewicht (Dutch)
adresse (French)	phone number	cost	poids (French)
dirección (Spanish)	admissions	regular price	peso (Spanish)
street address	tel	our price	waga (Polish)
alamat (Malay)	general phone	your price	peso con batteria in dotazione (Italian)
headquarters	general information	license	trọng lượng (Vietnamese)
adresa (Romanian)	telephone#	paperback	masa (Polish)
address 1	teléfono (Spanish)	hardcover	masa ciała (Polish)
mailing address	office	barnesandnoble	weight kg
town	information	us list price	vikt (Swedish)

The categorized results are shown in Table 4, together with an example in each category. The accuracy is 88%, and the false positives come from attributes with similar sets of values. For example, "overall" and "gameplay" from two game web sites are considered equivalent because both are for ratings of games.

The above experiment is about equivalence of attributes in data sources. We also check accuracy of synonyms for attribute name, which are independent of data sources. Table 5 shows the top 10 synonymous attribute names (ranked by similarity) of the four attribute names appearing in most data sources: "address", "phone", "price", "weight". We can see most synonyms have same or similar meanings with the original attribute.

4.4 Entity Equivalence

Besides equivalent attribute names, we also need to find equivalent entity names because users often use different entity names from those in our database. Equivalent entity names are generated by finding similar queries of each entity name. The query similarity is computed using a traditional approach in [8] and we briefly describe it here. We convert each query into a vector, by considering each URL as a dimension and the number of clicks on each URL as the value on that dimension. The similarity between two queries is just the cosine similarity between their vectors. We only consider pairs of queries with similarity at least 0.5 when finding equivalent names of entities. The equivalent entity names are precomputed and stored in an indexed database table. We uniformly randomly select 100 equivalent names of entities and manually label if they are truly equivalent, as shown in Table 6.

Table 6: Result of Equivalent Entity Detection

Two entities are	*Portion*	*Example*
Exactly same	87%	"australian job" vs. "job in australia"
One belongs to the other	7%	"flightless bird" vs. "large flightless bird"
One is a certain aspect of the other	5%	"will county" vs. "map of will county"
Different entities	1%	"1972 chevrolet suburban" vs. "1968 chevrolet suburban"

4.5 Data Retrieval and Result Aggregation

Given a web search query, FACTO generates every possible combination of entity e and attribute a. For each combination, it queries the database to get all values, together with URL of the page providing each value. FACTO also tries to replace a with every equivalent attribute name of a, or replace e with every equivalent entity name of e. For example, if a query is {britney birthdate}, FACTO will consider "britney" as an entity and "birthdate" as an attribute. Besides getting values for this combination, it will replace "birthdate" with "date of birth" and "dob", and "britney" with "britney spears". FACTO does not replace both entity and attribute at the same time, in order to avoid generating queries with different semantic meanings and to reduce computation cost.

After retrieving a set of values for a query, FACTO needs to select the best value as the answer and shows all URLs providing this answer. Among the values retrieved, some are for the original

entity e and attribute a, while some others are for an equivalent entity e' or equivalent attribute a'. Let $sim(e, e')$ and $sim(a, a')$ be the similarities between e and e' and that between a and a'. The goal of result aggregation is to select a value with highest confidence as the answer.

For each value v, FACTO stores the URL of the page containing v, and the StaticRank of that URL. (StaticRank is a measure for query-independent importance of a URL used by Bing Search, which is similar to PageRank.) The weight of a value v from URL u is defined as

$$w(v) = \begin{cases} StaticRank(u), \text{if } v \text{ is of } e \text{ and } a; \\ StaticRank(u) \cdot sim(e, e'), \text{if } v \text{ is of } e'; \\ StaticRank(u) \cdot sim(a, a'), \text{if } v \text{ is of } a'. \end{cases}$$

If we see the same value from different web domains[2], we are more confident that this value is correct. Some values may be similar to each other, which also improves the confidence of each of them being correct. Given all values $v_1, \dots, v_k$ from URLs $u_1, \dots, u_k$ for e and a, we define the score of each value v_i as

$$score(v_i) = w(v_i) + \sum_{\substack{j \in [1,k], \\ domain(u_i) \neq domain(u_j)}} w(v_j) \cdot sim(v_i, v_j)$$

Similarity between different values is defined based on their types. FACTO considers seven types of values: (1) string, (2) date/time, (3) numerical, (4) duration, (5) length, (6) area, (7) weight. For example, "6.4 lb" is parsed as a value of weight, and "July 1, 1978" as a value of date/time. A value is of type (1) if it does not belong any other type. The similarity between each type of values is defined based on our experiences with data[3].

After computing the score of each value, FACTO selects the value with highest score as the answer to the query. It also collects all values consistent with the selected value (having similarity no less than 0.9), and show these values when the user clicks "Show All".

4.6 Query Answering Experiments

Finally we study the performance of FACTO in answering web search queries, and compare that to the fact lookup engine of Google, and that of Ask.com which puts more focus on NLP-based question answering. We use a server with two Intel Xeon 2.66GHz Quad-core processors and 32GB memory as our database server, which runs Windows Server 2008 and SQL server 2008. Another server with the same configuration is used as web server. To test the performance on web search queries, we randomly sample 134K distinct queries from the query logs of Bing from Jan 2008 to March 2010, and the probability of a query being selected is proportional to its query frequency in this period. We send each query to FACTO, Google and Ask.com to retrieve the fact lookup answers when available. Examples of fact lookup answers of Google are shown in Figure 1, and that of Ask.com is shown in Figure 8. We crawled Google on 2010/08/10, Ask.com

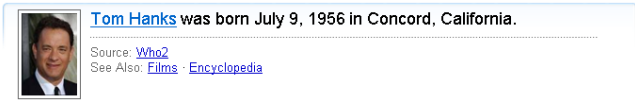


Figure 8: Fact lookup answer of Ask.com for query {Tom Hanks date of birth}

on 2010/08/20, and use the indexed documents of Bing on 2010/06/22 for FACTO. When tuning FACTO system, we can use other samples of queries (which should be used by Google and Asko.com as well), but not the query sample used for testing.

In order to judge the relevance of fact lookup answers accurately and consistently, a comprehensive set of guidelines are needed. First, we categorize the queries into five types as in Table 7, in which the first two types are fact lookup queries. We do not consider unit conversion queries and definition queries in our study, because it requires very different (and usually specially designed) approaches to answer them. If the intent of a query is not obvious, we infer that form the clicked URLs of Bing.

Table 7: Types of queries

Type	*Meaning*	*Example*
EA	The query asks for an aspect/attribute/fact about an entity	how old is george bush
F	The query is about a general fact without a particular entity	current prime interest rate
E	The query is about an entity without any specific aspect/attribute	full moon band
O	Other types of queries for which the user seeks answers or recommendations, including questions w\o definitive answers, definition queries and unit conversion queries	how to write a thesis paper
W	The direct answer should not be triggered because the user is not seeking any direct answer (e.g., navigational query)	download free movies

The following guidelines are strictly followed when evaluating the correctness of each direct answer. A correct answer must have the following three properties:

1. ***The answer is true or mostly true***. If the query asks for a simple fact, the answer should be true. If the query asks for a list of items, the answer should contain majority of items (if different items have similar importance) or the several most important ones. This is judged based on information at the time of crawling. (Some questions may not have a known definitive answer, such as {Tom Cruise height}. If an answer is popularly used in many websites, we consider it to be correct.)
2. ***The answer is consistent with search results***, i.e., the first result URL is about the same entity if the query contains an entity, or about the question being asked. In some rare cases the first result of a search engine is obviously irrelevant, we consider an answer to be consistent if it is consistent the most clicked result URL of Bing.
3. ***The answer is clear and specific***. It cannot contain too much irrelevant contents. However, it is OK to contain some related contents not being asked. For example, if a user searches for birth date of someone, and the answer contains birth place as well, it is still considered as relevant. An answer indicating "I don't know the answer" is considered as wrong answer.

If the intent of a query is not obvious and we cannot understand it from the result URLs of the corresponding search engine or Bing's clicked URLs, we say we cannot judge the correctness.

[2] A web domain is the domain name of a URL, such as "yahoo.co.jp" for http://blogs.yahoo.co.jp/kazami_0922.

[3] The similarity between two numerical values v_1 and v_2 is defined as $sim(v_1, v_2) = max\left(1 - 4 \cdot \frac{|v_1 - v_2|}{|v_1| + |v_2|}, 0\right)$. The similarity between values of type (4) to (7) is defined in the same way. The similarity between strings v_1 and v_2 is $sim(v_1, v_2) = max\left(1 - 4 \cdot \frac{EditDistance(v_1, v_2)}{v_1.length + v_2.length}, 0\right)$. The similarity between different date/time values is zero. The similarity between different types of values is zero.

Table 8 compares the precision and query coverage of fact lookup queries on Google, Ask.com, FACTO and FACTO without using equivalent attributes. We can see FACTO achieves higher precision and its query coverage is between Google and Ask.com. Detailed results are described below.

Table 8: Accuracy of fact lookup answers of Google, Ask.com, FACTO and FACTO w\o equivalent attributes

	Google	Ask.com	FACTO	FACTO w\o equv-attr
Precision	76.92%	65.90%	80.17%	84.47%
#correct answers	50	172	97	87

The accuracy of fact lookup answers of Google is shown in Table 9, where accuracy is defined by "#correct / (#correct + #wrong)". We also show the precision of fact lookup queries (EA+F), which is the number of fact lookup queries answered correctly divided by the number of queries triggering an answer that provides a fact. The false positives are caused by treating a non-fact-lookup query as a fact lookup query. Google has very few false positives, which shows it confines the triggering of its fact lookup answer very well. However, the value of its answer is sometimes inconsistent with the attribute. For example, for query {how old is george bush} it answers "george bush date of birth – 1946", although 1946 is not a date and the right answer should be "July 6, 1946". Majority of errors are simply incorrect answers, such as answering {ps3 release date} with "2010". We keep the result pages of Google with fact answers for our testing queries at http://lepton.research.microsoft.com/facto/doc/google_answer.zip.

Table 9: Accuracy of fact lookup answers of Google

Type	*#queries*	*correct*	*wrong*	*cannot judge*	*accuracy*
All	72	51	15	6	77.27%
EA	67	49	13	5	79.03%
F	1	1	0	0	100%
E	4	1	2	1	33.33%
O	0	0	0	0	N/A
W	0	0	0	0	N/A

fact lookup queries	*correct*	*wrong*	*false positive*	*precision*
	50	13	2	**76.92%**

The accuracy of fact lookup answers of Ask.com is shown in Table 10. Ask.com shows fact lookup answer more often than Google and FACTO, which might be the cause for its lower precision. For example, it answers {food new york mets stadium} with "The New York Mets play at Citi Field", which is not what the user asks for. It gives impressive results for some hard fact lookup queries. For example, for query {raven symone gives birth} it answers "Raven-Symoné is not and has never been pregnant according to reports", which shows it knows what has not happened besides what has. The result pages of Ask.com with fact answers can be accessed at http://lepton.research.microsoft.com/facto/doc/ask_answer.zip.

Table 10: Accuracy of fact lookup answers of Ask.com

Type	*#queries*	*correct*	*wrong*	*cannot judge*	*accuracy*
All	410	284	119	7	70.47%
EA	202	145	55	2	72.50%
F	34	27	6	1	81.82%
E	34	20	14	0	58.82%
O	105	92	9	4	91.09%
W	35	0	35	0	0%

fact lookup queries	*correct*	*wrong*	*false positive*	*precision*
	172	61	28	**65.90%**

Table 11 shows the accuracy of FACTO. FACTO provides correct answer more often than the other two, although it has some false positives. For example, it answers {british publisher} with the publisher of book "The British". We summarize the results of the three engines at http://lepton.research.microsoft.com/facto/doc/fact_answer_results.xlsx, and FACTO can be accessed at http://lepton.research.microsoft.com/facto/.

We also test the performance of FACTO without using equivalent entity names and that without using equivalent attribute names, as shown in Table 12 and Table 13. We can see equivalent entities are very important to FACTO, as both precision and query coverage drops when running without them. When not using equivalent attributes, precision increases and query coverage drops**.**

Table 11: Accuracy of fact lookup answers of FACTO

Type	*#queries*	*correct*	*wrong*	*cannot judge*	*accuracy*
All	124	97	24	3	80.17%
EA	117	97	17	3	85.09%
F	0	0	0	0	N/A
E	6	0	6	0	0%
O	1	0	1	0	0%
W	0	0	0	0	0%

fact lookup queries	*correct*	*wrong*	*false positive*	*precision*
	97	17	7	**80.17%**

Table 12: FACTO w\o equivalent entities

Type	*#queries*	*correct*	*wrong*	*cannot judge*	*accuracy*
All	92	71	19	2	78.89%
EA	88	71	15	2	82.56%

fact lookup queries	*correct*	*wrong*	*false positive*	*precision*
	71	15	4	**78.89%**

Table 13: FACTO w\o equivalent attributes

Type	*#queries*	*correct*	*wrong*	*cannot judge*	*accuracy*
All	104	87	16	1	84.47%
EA	99	87	11	1	88.78%

fact lookup queries	*correct*	*wrong*	*false positive*	*precision*
	87	11	5	**84.47%**

We analyze what kinds of errors are made by each engine. The errors are categorized as follows:

(1) Wrong answer, such as answering "English" for {Turkey language}.
(2) Incomplete answer, such as answering "2009" for {Diablo 3 release date}.
(3) Answer is not for the query, such as answering "santa monica college founded – 1929" for {how santa monica college was founded}.
(4) Entity query: The query is an entity and does not contain any attribute, such as {Microsoft publisher}.
(5) Navigational query, such as {Lil Wayne myspace}.
(6) Should not trigger answer for query, such as {watch free movies}.
(7) Answer indicates "I don't know", such as answering "Conversion from hours to miles is not available."

The types of errors made by Google, Ask.com and FACTO are shown in Table 14. We can see wrong answers do not account for majority of the errors, and many errors are caused by failing to

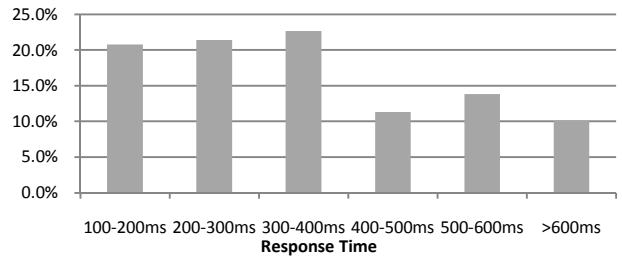


Figure 9: Response time of queries with answers

understand the query correctly. This suggests that query understanding is at least as challenging as information extraction.

Table 14: Types of errors by Google, Ask.com and FACTO

Type of error	*Google*	*Ask.com*	*FACTO*
Wrong answer	33.3%	5.9%	33.3%
Incomplete answer	40%	15.1%	16.7%
Answer is not for the query	13.3%	37.8%	37.5%
Entity query	6.7%	7.6%	8.3%
Navigational query	6.7%	12.6%	4.2%
Should not trigger answer	0	14.3%	0
Answer "I don't know"	0	6.7%	0

The average response time of FACTO is 10.6ms for every query, and 381ms for every query with an answer returned. Figure 9 shows the distribution of response time for queries with answers.

We perform Student's t-test on the precision and coverage of the three engines plus FACTO w\o equivalent attributes (denoted by FACTO*). Table 15 contains the results, which shows the probability of FACTO having higher precision than Google is 0.695, and that against Ask.com is 0.999. The query coverage of the fact lookup engines are in the range of 0.05% to 0.2%, which actually correspond to a large number of queries considering there are >4B searches per day. We also believe fact lookup answers will gradually change users' habits from searching for web pages to searching for facts.

Table 15: T-test on accuracy/coverage of different engines. Each cell in upper-right part contains the probability of engine of the row having higher precision than engine of the column. Lower-left part contains that for coverage.

One-tail probability	*FACTO*	*FACTO**	*Google*	*Ask.com*	
FACTO		0.200	0.695	0.999	*Precision*
*FACTO**	0.231		0.882	1.000	
Google	0.000	0.001		0.966	
Ask.com	1.000	1.000	1.000		
	Coverage				

5. CONCLUSIONS AND FUTURE WORK

In this paper we describe FACTO, an end-to-end system for answering fact lookup queries with comparable performance with the fact lookup engines of popular search engines. Our future plan includes extracting more types of data to increase query coverage and precision, considering the temporal aspects of extracted information to provide more timely answers, and performing deep analysis on the data to improve data quality.

6. REFERENCES

[1] E. Agichtein, L. Gravano. Snowball: extracting relations from large plain-text collections. *Digital Libraries'00*.

[2] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead and O. Etzioni. Open information extraction from the Web. *IJCAI'07*.

[3] E. Brill, S. Dumais and M. Banko. An analysis of the AskMSR question-answering system. In *EMNLP'02*.

[4] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Uncovering the relational web. *WebDB'08*.

[5] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: Exploring the power of tables on the Web. *VLDB'08*.

[6] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001. www.csie.ntu.edu.tw/~cjlin/libsvm

[7] E. Crestan and P. Pantel. Web-scale knowledge extraction from semi-structured tables. *WWW'10*.

[8] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. *WWW'02*.

[9] 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). *WWW'04*.

[10] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krupl, and B. Pollak. Towards domain-independent information extraction from web tables. *WWW'07*.

[11] S. M. Harabagiu, M. A. Paşca, S. J. Maiorano. Experiments with open-domain textual question answering. *COLING'00*.

[12] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *ACM SIGOPS Operating Systems Review*, 41(3):59-72, 2007.

[13] J. Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5), pp 604-632, 1999.

[14] X. Li. Understanding the semantic structure of noun phrase queries. *ACL'10*.

[15] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. E. Moser. Extracting data records from the web using tag path clustering. *WWW'09*.

[16] M. Paşca, D. Lin, J. Bigham, A. Lifchits, A. Jain. Organizing and searching the World Wide Web of facts - step one: the one-million fact extraction challenge. *AAAI'06*.

[17] John Prager. Open-domain question-answering. *Foundations and Trends in Information Retrieval*: Vol. 1: No 2, pp 91-231, 2006.

[18] S. Tellex, B. Katz, J. Lin, A. Fernandes, G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. *SIGIR'03*.

[19] A. Tengli, Y. Yang, and N. L. Ma. Learning table extraction from examples. *COLING'04*.

[20] Y. Wang and J. Hu. A machine learning based approach for table detection on the web. *WWW'02*.

[21] A. Yates, M. J. Cafarella, M. Banko, O. Etzioni, M. Broadhead and S. Soderland. TextRunner: open information extraction on the web. *NAACL-HLT'07*.

[22] X. Yin, W. Tan, X. Li and Y.-C. Tu. Automatic extraction of clickable structured Web contents for name Entity queries. *WWW'10*.

[23] X. Zhou, J. Gaugaz, W.-T. Balke and W. Nejdl. Query relaxation using malleable schemas. *SIGMOD'07*.