QuTE: Answering Quantity Queries from Web Tables

Vinh Thinh Ho  
hvthinh@mpi-inf.mpg.de  
Max Planck Institut für Informatik  
Saarbrücken, Germany

Koninika Pal  
kpal@mpi-inf.mpg.de  
Max Planck Institut für Informatik  
Saarbrücken, Germany

Gerhard Weikum  
weikum@mpi-inf.mpg.de  
Max Planck Institut für Informatik  
Saarbrücken, Germany

ABSTRACT
Quantities are financial, technological, physical and other measures that denote relevant properties of entities, such as revenue of companies, energy efficiency of cars or distance and brightness of stars and galaxies. Queries with filter conditions on quantities are an important building block for downstream analytics, and pose challenges when the content of interest is spread across a huge number of web tables and other ad-hoc datasets. Search engines support quantity lookups, but largely fail on quantity filters. The QuTE system presented in this paper aims to overcome these problems. It comprises methods for automatically extracting entity-quantity facts from web tables, as well as methods for online query processing, with new techniques for query matching and answer ranking.

ACM Reference Format:

1 INTRODUCTION
Motivation. Searching for entities with filter conditions on quantitative properties, like financial, physical or technological measures, is common in database applications and well supported. For Internet contents, including text and tables in web pages, however, search engines often fail to provide answers as they do not understand quantity values and their units and do not infer which quantity refers to which entity. The following are example queries that an analyst or data scientist could pose:

- British football teams worth more than 1 Billion Euros
- cities with annual energy consumption above 50 TWh
- European football stadiums with more than 60000 seats

For such queries, search engines sometimes return pages with largest/smallestrichest/etc. entities, such as Wikipedia lists, but this depends on the keywords and the quantity value in the query. For example, “celebrities worth more than 1 billion” returns a list, but the seemingly minor variations “…above 1.5 billion” or “women worth…” fall back to merely giving ten blue links to all kinds of web pages – some off topic, some worthwhile reading but still needing human effort to identify answer entities within long documents. Note that these kinds of queries with quantity filters are different from quantity lookups, such as “net worth of Rihanna”. Lookups are much simpler and handled fairly well by search engines and QA assistants; they do not involve the interpretation of filter conditions.

Structured data and knowledge bases, accessible through web APIs, could be considered as a solution. However, the relevant databases are scattered across the Internet, posing obstacles in discovering them and understanding their schemas, options for joins and unions, trustworthiness, etc. Knowledge bases, such as Wikidata, have good coverage of entities, but barely contain quantitative properties (e.g., 2000 marathon runners but only 20 with their best time captured, 3000 car models but none with engine power, energy efficiency, carbon footprint, etc.).

Approach. The research that led to this system demo proposal pursues the goal of supporting web-based quantity queries, bringing database functionality to unstructured and semi-structured content. Our work includes tapping into textual contents [5, 6] as well as web tables (embedded in HTML pages) [7]. This demo paper is about the latter.

We present the QuTE system, which taps into web tables to answer quantity queries. It has two major components: 1) extracting entity-quantity facts from tables using machine-learning techniques, and 2) matching queries against an indexed repository of such facts and computing ranked answer lists. The extraction component is described in detail in [7]. The query processor is based on techniques originally developed in [5] for the different case of quantities in text documents. For the case of web tables, we devised several extensions, in particular, for contextualizing quantity facts and for enriching them with evidence from text corpora to enhance the answer ranking. This demo largely focuses on the query processing part. The QuTE system can be accessed at https://qsearch.mpi-inf.mpg.de/table/.

2 SYSTEM OVERVIEW
We refer to the entity-quantity facts that QuTE taps into as Qfacts: triples of the form (entity, quantity, context) where the context consists of words and phrases that add evidence and explanation to the fact (e.g., (Anfield, 61905, “record attendance 1952”)). The two main stages of QuTE are the extraction of Qfacts from web tables, carried out as an offline computation, and the online query processing. The first stage comprises components for entity linking, quantity normalization, column alignment, and Qfact contextualization. The second stage involves matching and ranking of Qfacts against user queries. Figure 1 gives a pictorial overview.

Entity Linking. The processing of web tables starts with detecting entities and quantities. These are standard tasks for text analysis. Entities are disambiguated by linking them to the Yago knowledge graph, by inference based on a probabilistic factor graph (similar to [2]).
Table 1: Example table: entities and quantities are in light green and red, respectively; arrows indicate the column alignment.

<table>
<thead>
<tr>
<th>Team</th>
<th>Stadium</th>
<th>Capacity</th>
<th>Coach</th>
<th>Value (in Bio)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayern</td>
<td>Allianz Arena</td>
<td>81,044</td>
<td>Hansi Flick</td>
<td>2,549 Euro</td>
</tr>
<tr>
<td>Real</td>
<td>unknown</td>
<td>n/a</td>
<td>Pep Guardiola</td>
<td>2,055 GBP</td>
</tr>
<tr>
<td>Man City</td>
<td>unknown</td>
<td>n/a</td>
<td>Zinedine Zidane</td>
<td>3,649e9; £</td>
</tr>
<tr>
<td>Chelsea</td>
<td>Stamford Bridge</td>
<td>40,834</td>
<td>Frank Lampard</td>
<td>1,958 GBP</td>
</tr>
</tbody>
</table>

### Quantity Normalization

This step involves detecting numerical expressions and inferring their units when applicable, e.g., from column headers. We build on rule-based prior works [12, 13], with some extensions. We can now easily classify table columns into E-columns (mostly rows with entities), Q-columns (mostly quantities) and Others (e.g., text comments).

### Column Alignment

For meaningful Qfacts, we need to go beyond the above extraction steps, and compute which quantity denotes a property of which entity. This is challenging for complex tables with more than one E-column. Table 1 shows an example. In [7] we have devised a novel solution for this column alignment problem, with much higher precision than prior baselines [3, 11, 18]. In a nutshell, we compute co-occurrence scores for entity-quantity pairs for candidate alignments, aggregating over the rows of the two columns. The co-occurrences are obtained from proximity-based matching in a large text corpus (Wikipedia articles and other web content). Details are in [7].

### Contextualization

Bare Qfacts are not sufficient for effective search, as user queries often contain cues about the measure of interest and all kinds of modifiers. For example, the query “British football teams worth more than 1 Billion Euros” includes the measure “worth” and the modifier “British”. Therefore, we augment each of the extracted entity-quantity pairs with context words derived from the column headers, the table caption, the other values in the same row, the page title, the labels on the DOM-tree path to the table (e.g., intermediate headings) and table surrounding text.

### Query Matching

Queries are decomposed into the answer type (e.g., football teams), the quantity condition (e.g., ≥ 1,000,000,000 Euro) and contextual cues. These three components are matched against the (millions of) Qfacts from the extraction stage. For the context word matching, we use IR-style language models with word embeddings, so that semantically related matches are captured as well (e.g., “worth” matching “value”). Section 3 gives more detail.

### Answer Ranking

The ranking of the returned answers is mostly based on the text-matching scores for the query and Qfact contexts. In addition, we devised two novel components that provide signals for corroboration and enhanced ranking. First, we identify witnesses for the confidence in a Qfact answer, in the form of evidence snippets from a large corpus of Wikipedia articles and web pages. These are automatically retrieved by contextualized search, at query runtime. Second, we compute a score for the mutual consistency of candidate answers, reflecting their agreement on quantity values in the same order of magnitude and on related context words. These two kinds of signals are added to the context-matching scores with hyper-parameter weights.

## 3 QUERY MATCHING

Queries are matched against the repository of Qfacts by 1) testing that entities belong to the semantic type stated in the query (using the rich type system of the Yago knowledge base), 2) evaluating the quantity filter condition, and 3) matching all cue words in the query against the precomputed context of the candidate Qfacts. The Qfact context is obtained from six regions of the web page:

1. headers of the E-column and Q-column where the fact appeared
2. the page title
3. labels in the DOM-tree path to the table
4. the table caption
5. all other cells of the same table row
6. text surrounding the table, within a window of 100 words

The QuTE demo highlights all of these contextual cues. Extensive experiments, with tuning of hyper-parameter weights, showed that the regions 1 and 2 are by far the most valuable [7]. For the regions 3 to 6, the noise-to-signal ratio is often high.

For the degree of matching query cues Q against Qfact context X, QuTE employs a language-model-based scoring function. This incorporates word2vec embeddings to capture the influence of semantically related words (e.g., “worth” and “value” being near-matches). Specifically, we use the following weighted directed embedding distance (wded), originally proposed in [5] and adapted to the new setting of web tables:

\[
\text{wded}(Q, X) = \frac{\sum_{u \in Q} \sum_{v \in X} \omega(u) \cdot \min_{v \in X} (d(u, v) \cdot L(v))}{\sum_{u \in Q} \omega(u)}
\]

where \(\omega\) are tf-idf weights of tokens, \(d(u, v)\) is the cosine between word embeddings, and \(L(v)\) denotes the weight of token \(v\) depending on which region it is collected from.

## 4 ANSWER RANKING

The ranking of candidate answers is primarily computed by the matching score (see Section 3). In addition, we harness two unconventional signals to further enhance the ranking.

### Evidence from Text Corpus

As the context of table-based Qfacts is often sparse or noisy, QuTE has a new technique of collecting...
witnesses for Qfact validity by retrieving evidence from a large text corpus of Wikipedia articles and other pages from a large news corpus, a total of 13.5 Million documents.

This evidence collection is performed for each answer candidate \( A \) at query time. We generate a keyword-style search over the text corpus based on the cue words in the query and the most important context words of the Qfact (i.e., column headers and page title), plus the entity and quantity of the Qfact. Ideally, we would find witnesses \( W \) with all these constituents, but we expect and handle also partial matches (as is the nature of keyword search) and approximate matches for the quantity value (e.g., rounded values or values within an interval, but ideally with the same unit). These considerations are cast into an evidence-score \((A, W)\):

\[
\left( w_1 \cdot \text{sim}_1(A.\text{cxt}, W) + w_2 \cdot \text{sim}_2(A.\text{qty}, W.\text{qty}) \right) \cdot \text{sim}_3(A.\text{ent}, W.\text{ent})
\]

As some entities of the table-based Qfact repository may not appear in the text corpus at all, or merely in sophisticated text where they are hard to spot, we extend the search to other entities that match the query structure: same type, similar context cues, same quantity measure (disregarding the value). These same-type matches are viewed as confirming the proper query interpretation. Perfect matches of the Qfact entity itself carry higher weight than finding same-column entities; this is reflected by the entity-similarity weights in the above scoring function.

Top evidence scores per witness are averaged into an overall score for each candidate answer. The QuTE demo allows exploring the witnesses at query time. It involves additional text search and computation, but usually runs in a few seconds (after clicking on the “Evidence” button in the query result page).

Mutual Consistency of Answers. For a given query, we expect different answers to be consistent in two ways:
1. Their context cues are similar. For example, “worth”, “value”, “investment”, etc. are consistent cues, but “salary”, “taxes” etc. would be outliers for a query about the value of football teams.
2. Their quantity values should have limited variance. For example, team values of football clubs should be in the millions or billions, but not in the thousands or trillions.

To compute a consistency score, we adapt a randomized cross-validation technique, originally proposed for image classification [17], which is completely self-supervised. Details are in [7]. The QuTE demo supports different configurations for this component.

Putting Everything Together. The three signals – matching score, evidence witnesses and mutual consistency – are combined by a weighted sum, with weights tuned via a small development set of gold-standard query-answer pairs.

5 DEMONSTRATION

QuTE is accessible at [https://qsearch.mpi-inf.mpg.de/table/](https://qsearch.mpi-inf.mpg.de/table/). Figure 2 shows a screenshot of top-ranked results for the example query “sprinters who ran 100 meters under 9.9s”.

Data Sources. QuTE runs on a repository of ca. 18M Qfacts for ca. 800K entities, extracted from two large corpora: WikiTables consisting of 1.8M tables from English Wikipedia, and TableL [8] consisting of 2.6M tables from 1.5M Common Crawl web pages.

The text-based witnesses draw from a text corpus of ca. 13.5M documents from Wikipedia and news articles.

Input. QuTE accepts user input in two modes: form-based and free-text. In form-based mode, users enter a query into three text fields (entity-type, quantity-filter, context-cues), with auto-completion suggestions for types. This mode helps our engine to evaluate the query as precisely as possible. In free-text mode, user queries are telegraphic phrases or full-fledged questions, which are then decomposed into the three constituents by a rule-based parser, as shown in the demo. The parser uses the Yago type taxonomy and a dictionary of quantity units.

To switch between the two input modes, users click on the toggle button on the left side of the search bar.

Output. QuTE computes a ranked list of entity answers, based on their confidence scores (see Section 4). The parsed query and top-k answers are shown to the users, along with context cues for each answer. Different colors are used to highlight answer entity (in light green), answer quantity (in red), column headers (obvious), page title (in orange), DOM-tree headings (in orange), table caption (in green), table row (obvious) and a surrounding-text snippet (in cyan), when available. By default, QuTE shows only the highest-scoring Qfact for each distinct entity. By clicking on the “Show more” button, additional Qfacts for the same entity can be displayed. These often include a noisy tail, showing the difficulty of the task.

Table 2 shows more anecdotal examples of quantity queries and their top answers produced by QuTE. For comparison, the table also shows top-3 results from a major search engine (SE), with incorrect, uninformative or very partial results in purple. These are evidence that search engines do not properly interpret quantity queries and simply return string-level matches. Some are high-quality list pages: browsing through them would give the user good results, but this...
Table 2: Queries with top-3 answers by QuTE vs. search engine (as of March 5, 2021). Purple answers are uninformative or very partial.

<table>
<thead>
<tr>
<th>Query: Buildings higher than 500 meters</th>
<th>QuTE</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Burj Khalifa, One World Trade Center, Taipei 101, etc.</td>
<td>List of tallest structures – 400 to 500 metres - Wikipedia, List of tallest structures - Wikipedia, China’s skyscrapers taller than 500 meters - Global Times</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query: Which universities have more than 60000 students</th>
<th>QuTE</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liberty University, University of Belgrade, University of Toronto, etc.</td>
<td>List of United States public university campuses by enrollment, The top 50 US colleges that pay off the most 2019, There Are Now 50 Colleges That Charge More Than $60000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query: Stadiums with more than 100000 seats</th>
<th>QuTE</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Melbourne Cricket Ground, Kiangre 1st of May Stadium, Tiger Stadium (LSU), etc.</td>
<td>List of Eurostadiums by capacity - Wikipedia, World’s Largest Sport Stadiums - Topend Sports</td>
<td></td>
</tr>
</tbody>
</table>

Further options are to customize the settings for column alignment and for consistency-based corroborations (see Section 4 and [7]).

By default, QuTE answers are ranked by confidence score. Users can re-rank them in ascending or descending order of quantity value, or by entity prominence via clicking on the “Sort by” button.

**Limitations and Outlook.** QuTE brings database-like queries to web contents: functionality that would be easy to express in SQL if the data resided in a single DB, but is novel for ad-hoc web tables. Being a prototype system, QuTE inevitably has notable limitations. The query parser for decomposing user input in free-text mode uses rules that may fail on certain phrases or miss out on certain kinds of quantities. The form-based input mode could possibly alleviate these issues, but is far from perfect either. We plan to enhance QuTE by expanding its library of measures and units, and by adding awareness of temporal scoping.

Another limitation is that QuTE can handle only queries with one quantity filter. For example, “stadiums with more than 100k seats built after 2000” is beyond the current scope. We are working on extensions, to address such queries as well as support for group-by (e.g., “100m sprints with more than 5 races under 9.9s”).

**6 RELATED WORK**

Entity-centric knowledge extraction from web tables has been intensively explored (see, e.g., [2, 3, 9, 11, 18]). However, a common assumption has been that each table has a single subject column to which all other columns refer. This does not hold for complex tables such as the example in Table 1. Moreover, to support search over quantity properties, the bare extraction is insufficient, as it lacks contextualization with cue words that relate to user queries.

Methods for detecting numerical expressions (with units) have been developed [1, 12, 13]. However, inferring to which entity a detected quantity refers in a complex table row or sophisticated text passage, is out of scope for these methods. Our recent works [5, 7] close this gap.

Question answering over tables has been addressed from different angles, like translation into structured queries, machine learning for answer prediction, linkage with knowledge bases, and more (e.g., [4, 10, 13–16]). Closest to our work is [13], which has limited support for quantity filters, though.
REFERENCES


