Timeline Generation for Fluent Quantities

M. Tech. Dissertation

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

by

Abhirut Gupta

Roll No: 123050016

under the guidance of

Prof. Sunita Sarawagi

Department of Computer Science and Engineering

Indian Institute of Technology, Bombay

Mumbai

Dissertation Approval Certificate

Department of Computer Science and Engineering

Indian Institute of Technology, Bombay

The dissertation entitled **"Timeline Generation for Fluent Quantities"**, submitted by **Abhirut Gupta** (Roll No: **123050016**) is approved for the degree of **Master of Technology** in **Computer Science and Engineering** from **Indian Institute of Technology, Bombay**.

Prof. Sunita Sarawagi
Dept. of CSE, IIT Bombay
Supervisor

Prof. Ganesh Ramakrishnan
Dept. of CSE, IIT Bombay
Internal Examiner

Dr. Muthusamy Chelliah
Yahoo! India
External Examiner

Prof. S. Sudarshan
Dept. of CSE, IIT Bombay
Chairperson

Place: IIT Bombay, Mumbai
Date: 27^{th} June, 2014

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Abhirut Gupta

Signature

ABHIRUT GUPTA

Name of Student

123050016

Roll number

27/06/2013

Date

Abstract

Today, Information Retrieval seems to be moving in a new direction. Instead of the old scheme of returning a list of most relevant documents in response to queries, the focus is shifted to answering user's information need is a structured manner so as to reduce the cognitive burden on a user as much as possible. Here we present QTQ, a system which generates a timeline in response to queries seeking fluent quantities. We use web pages as our data source, and the inherent noise presents a formidable challenge for information extraction. We propose a novel approach, which combines extractions from both free text and tables found on web pages. Instead of making hard extraction decisions, we keep at hand various alternative extractions with associated confidence scores, and use a consensus model to collectively score candidates. Our experiments clearly support the gains of using a collective model versus independent extractions. One of the main parts of the collective model is the temporal regression model, which models the distribution of values for each time epoch. Given the noisy nature of extractions from the web, we use a Kernel Density model for the task. Experiments on our dataset comprising of about 270 queries, show that the Kernel Density model outperforms the Gaussian Processes baseline by 11% for MAP and by 78% for average probability of the correct answer over all gold time values. The collective method outperforms the independent method by 20% for map and by 32% for probability of the correct answer.

Acknowledgements

I would like to express my sincere gratitude to **Prof. Sunita Sarawagi** for her guidance throughout the course of the project. I would like to thank Ankit Gupta, Mandar Joshi, Piyush Dungarwal and Vinita Sharma for their continued support and helpful discussions.

I would also like to thank my parents, for their unwavering support and encouragement as I worked towards my degree.

Contents

List of Figures

Chapter 1

Introduction

In recent years, a few systems [Wu and Marian, 2007, Banerjee et al., 2009a, Bakalov et al., 2011, Sarawagi and Chakrabarti, 2014] have been proposed for responding to queries seeking quantities. While some quantities (value of π, number of sides in a heptagon) have no uncertainty about them (although measurements may always be approximate), more common in Web search are quantities wth substantial uncertainty and/or variability. These may arise from many different root causes, some of which are listed below:

- Ambiguity: Asking for the frequency of cordless phones or the half-life of Plutonium is ambiguous; there are multiple standard frequencies and isotopes of Plutonium.
- Population diversity: The battery life of an iPad or the height of a giraffe is a distribution over many instances in a population of iPads and giraffes.
- Temporal variation: Atmospheric CO_2, population of countries, net worth of a person, revenue of a company, all change with time.

1.1 Our Goal

We focus exclusively on temporal variation. Specifically, we describe a system called QTQ (querying for temporal quantities) that accepts a two-part query consisting of textual descriptions of an entity (e.g., `bill gates`) and an attribute (e.g., `worth`), with an optional unit (e.g., `dollars`), and then automatically searches HTML tables and text on the Web to discover and report quantity responses along a timeline. A sample output for the query, {"Finland",

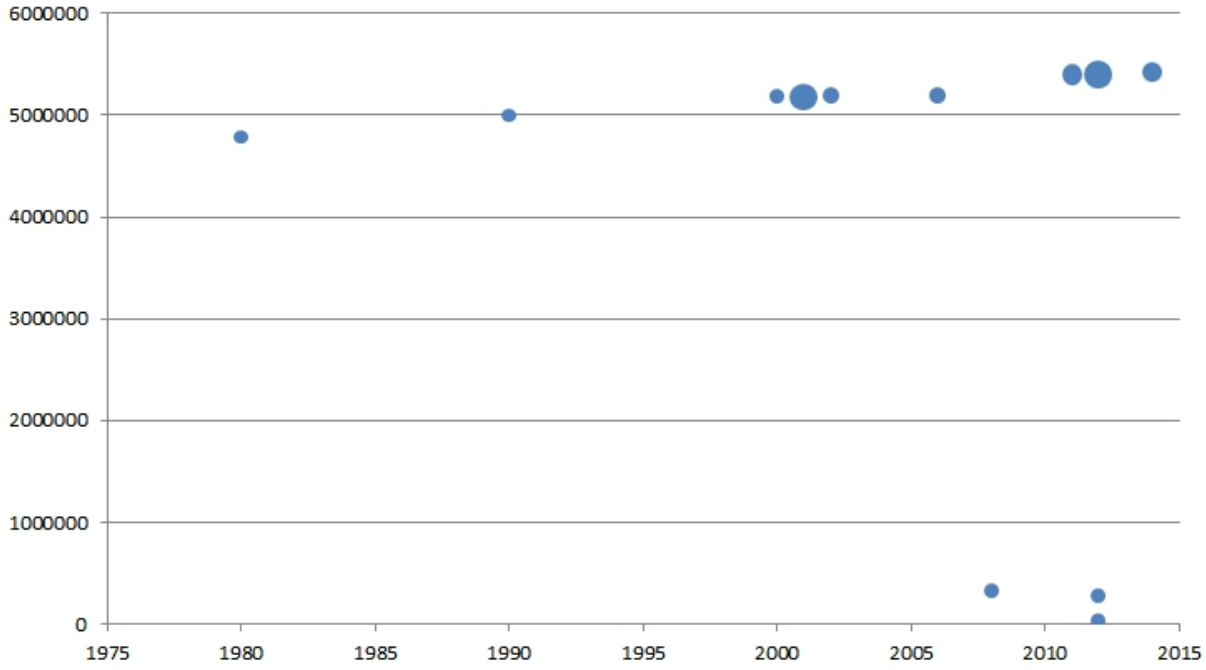

Figure 1.1: Top few extractions for the query - “Finland Population”

“Population”, null} is shown in Figure 1.1. The size of various points is proportional to their final confidence from our collective model.

1.2 Challenges

The above goals of QTQ are significantly different and extended compared to prior work [Banerjee et al., 2009a, Bakalov et al., 2011, Sarawagi and Chakrabarti, 2014], and highly non-trivial. Quantities and units are expressed in myriad surface forms in unstructured text and tables. The same quantity type may be expressed in diverse units. A quantity may be reported to different approximations in different source documents. Disambiguating units from tokens like “m” or “l” is difficult. Clues to the entity or attribute often lurk in the textual contexts of tables. Most important, when a quantity shows temporal variation, it has to be discovered and teased apart from the other forms of variability listed above. As a vital step, each quantity extracted has to be (probabilistically) associated with a time epoch, if evidenced near the extraction site.

Figure 1.2 presents sample extractions from text and tables for the query {”India”, ”Population”, ”kiloton”}. We shall use this example to highlight a few challenges of the task.

- In the text, *t1*, the quantity of interest (in bold) is lexically separated from the unit “kiloton”.

- The unit is present as "kt" which is ambiguous, and can also be short for the unit "carat(purity)".
- There are multiple temporal expressions present in vicinity of the quantity of interest *(2009, 1960)*.
- The quantity extracted from this snippet, is the value of the attribute "CO2 emission from liquid fuel consumption" and hence, not a correct answer to the query. The confidence score for such an extraction must be low in the final answer.
- In the table, *t2*, the relevant temporal expression is present in one of the many context texts present in the vicinity of the table.

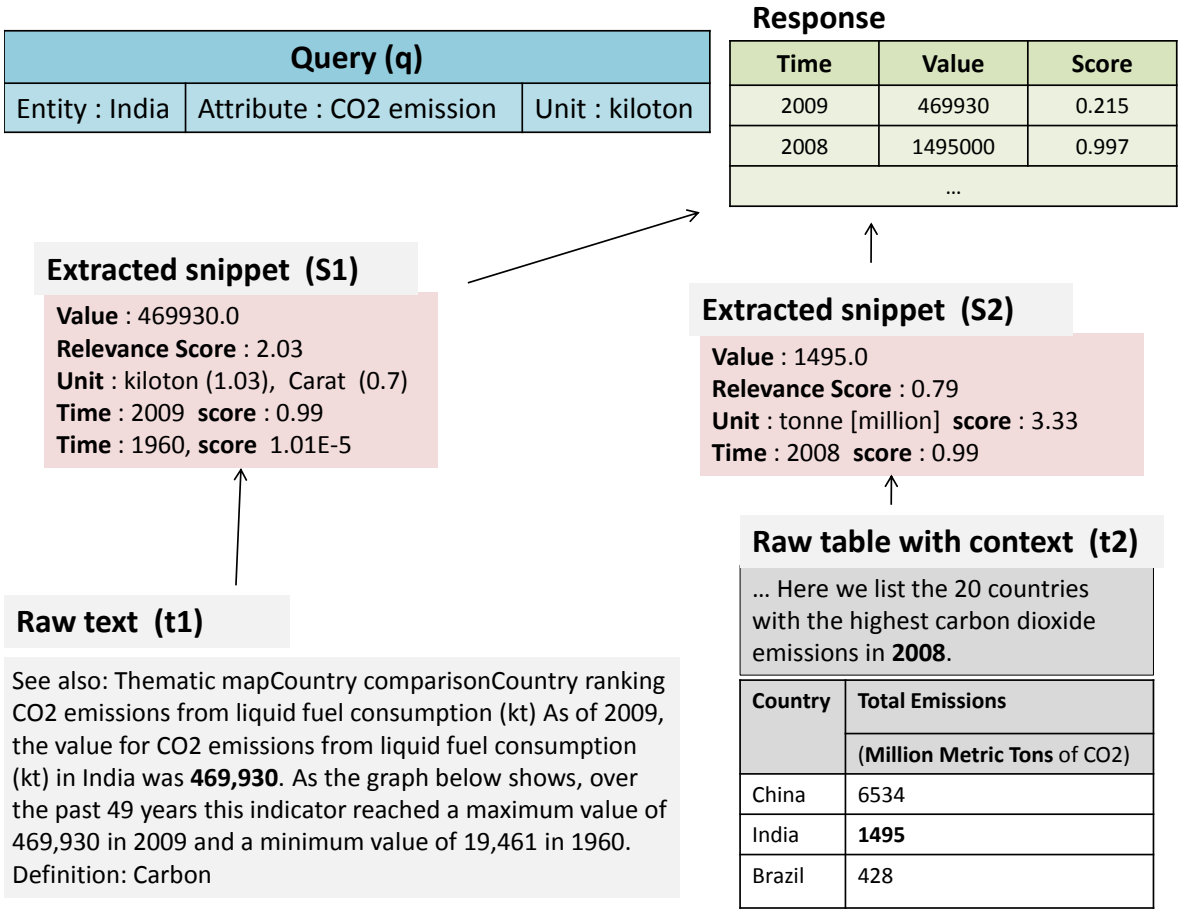

Figure 1.2: An example query, the raw text and table sources, and the returned response.

To our knowledge, QTQ is the first system satisfying all the above requirements. Apart from introducing and formalizing the problem, we present critical algorithmic building-blocks and their evaluation:

- Choice of the temporal model for open-domain Web-supported queries.

- Methods of extracting candidate values, their units, and temporal evidence from text and tables.
- Collective (across extraction sites) inference for generating timeline responses.

1.3 Organization of the Report

The rest of the report is organized as follows. We briefly discuss related work in the area in Chapter 2. Chapter 3 details the high-level design of QTQ. Candidate extraction, temporal and unit extraction and association and other pre-processing steps are detailed in Chapter 4. Chapter 5 presents the overall collective model, which refine scores of individual noisy extractions, and its components. In Chapter 6 we present a detailed evaluation of our system under various query loads. The report concludes with a discussion of the direction of future work in Chapter 7.

Chapter 2

Related work

The problem of generating timelines for temporally varying numerical attributes has not been addressed before to the best of our knowledge.

Quantity search without temporality has been studied before. Moriceau [Moriceau, 2006] was among the earliest to formalize quantity search, and provide some initial notions of temporal trends and aggregation of values. A more extensive system was built by Wu and Marian [Wu and Marian, 2007, "W&M"]. Banerjee etal [Banerjee et al., 2009a, "QCQ"] proposed the quantity interval ranking problem. SCAD [Bakalov et al., 2011] collected quantities while satisfying domain-guided numeric constraints between them (e.g., a laptop screen is wider than it is tall). All of these works ignore time.

Recently, [Zhang and Chakrabarti, 2013] propose to extract numerical attributes only from Web tables as valid at a particular point in time. Extraction from text documents is significantly more challenging because of more potential for irrelevant answers. Also, They do not model uncertain value distributions. Their aggregation/consensus is based on exact match of values, which we demonstrate as weaker than our value distribution model. Their "query" resembles a table completion task, with $\sim$100 entities, time, units, and scales explicitly provided. In contrast, we generate the timeline given only an entity and attribute name and do not require explicit specification of a unit.

[McClosky and Manning, 2012] present a proposal for generation of temporal ranges for facts that are valid only within a specific date. This problem is very different from ours, because each fact has a start and end date. In our case, the attribute itself is valid forever but it just takes different values at different points in time, and the value is quantitative.

Chapter 3

Architecture

Figure 3.1 shows the architecture of QTQ. We present an overview of the main components next. The user query q comprising of an entity string e_q (e.g. "Argentina", "Microsoft"), a time-varying attribute name string a_q (e.g. "Forest area, "Revenue"), and an optional unit name (e.g. "square kilometer", "US dollars") is submitted to QTQ. The query words are used to collect text documents from an indexed collection of Web documents (e.g. Google's) and an indexed store of Web tables (e.g. WWT's [Pimplikar and Sarawagi, 2012], or Web tables at research.google.com/tables.). Figure 1.2 shows examples of raw text and tables that we get in response to a query. Converting these raw sources into the answer in the form of time-value distribution involves several non-trivial extractions and aggregations. We briefly summarize these main steps next.

3.1 Snippet generation

A snippet is centred around a quantity that is deemed to be the value of the attribute at a given point in time. The context in which the quantity is expressed is processed to estimate the relevance of the quantity to the query (r_{sq} or r_s, where $r_s \in [0, 1]$), the unit of the quantity, and the time at which quantity is valid. Naturally, the definition and representation of snippets is very different for plain text and tables.

3.1.1 Table snippet

A table snippet is centred around a cell in a table that contains the quantity. For example, in the table, *t2*, in Figure 1.2, the cell at row 2 and column 1 is a snippet for the query {"India",

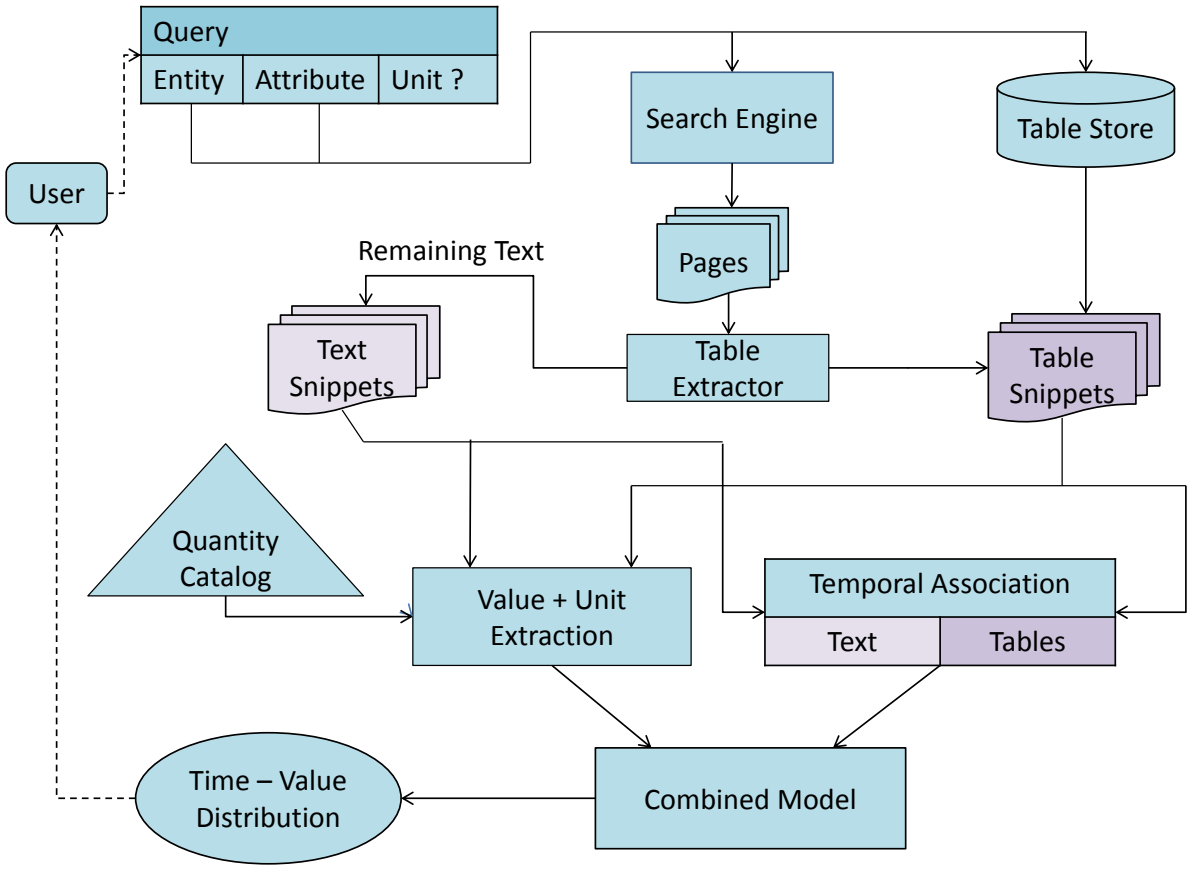

Figure 3.1: System architecture.

.. world wide prices for Mars Bars. Price of a regular sized bar (~60g) is **1 50** (USD) in United States, Caolifornia. The chocolate bar ..

Figure 3.2: Sample Snippet

"CO2 emission"}. In general, a table can generate more than one snippet. A cell (r, c) at a row r and column c of a table is a candidate snippet, if it contains a quantity, the entity string e_q matches either a cell in the same row, or the table is established to be about e_q, the attribute string q_q matches a header of column c or the text surrounding the table. All such matches can be quite noisy, therefore the relevance value of the snippet to the query has a lot of uncertainty associated with it. The exact method of established this relevance and generating the candidate snippets can be found in [Sarawagi and Chakrabarti, 2014].

3.1.2 Text snippet

A text snippet is centred around quantities found in the vicinity of entity and attribute words of the query. In Figure 1.2, one of the snippets created from the raw text, *t1*, is *s1*. For obvious reasons, this simplistic method of creating candidates from text on the web leads to many noisy extractions. We therefore apply many hand crafted rules to filter out such noise. Though, some useful information is lost, but the process leads to better quality of snippets retained.

The snippets are then ranked on their relevance to the query using a supervised RankSVM model as presented in [Joachims, 2002]. The features are tf-idf based for query words in the snippet. Details of the feature set and learning can be found in [Banerjee et al., 2009a]. The ranking score, is used as the relevance score (r_s), after putting it through a sigmoid function and multiplying with a parameter θ. The parameter θ is empirically determined, and serves to adjust the relative importance of text and table relevance scores.

3.2 Quantity and Unit Extraction

We extract from each snippet, the possible values of the quantity by parsing the noisy string for the different numbers across different locales that it can represent. For example, in Figure 3.2 the quantity in the snippet "1 50" can be parsed to extract two different float values - 1.5 and 150.

The values are meaningless without their associated units. Typically, the unit is mentioned

in a noisy format in the context surrounding the quantity. We identify and annotate this mention with respect to a unit in a well-established catalog of units. Specifically, we use the QuTree catalog that contains a list of 750 well-established units of measurements grouped into 44 quantity types, such as Length, Area, and Speed. Each quantity type has a *canonical* unit, and other units come with conversion factors to/from the canonical unit.

The extraction of quantities is extremely challenging in both table and text. For tables, the unit can be present either in the cell containing the quantity, or the header of the quantity column or in rare cases split across the two. The correct extraction of the units is an extremely challenging task and [Sarawagi and Chakrabarti, 2014] presents a context free grammar based parser for performing this extraction task. For text, the unit string is typically present right after the value (e.g. 50 kmph) or right before the value ($ 100), but sometimes the unit can be far from the quantity as shown in the text for snippet *s1* where the unit "kt" is separated from the quantity "469,930". We modify unit tagger described in [Sarawagi and Chakrabarti, 2014] to extract units from free format text. The quantity tagger outputs a list of units with their scores: (u_{s1}, g_1),...,(u_{sj}, g_j). Each unit u_{sj} with its type t_{sj}, gives a different value of the quantity, v_{sj}.

3.3 Temporal clues extraction and association

Next, we process each snippet to identify temporal clues that can help establish the date at which the quantity is stated in the snippet. We solve this problem in two steps: first we identify all time expressions in the context around the quantity, next we score each candidate time with a real-value indicating the degree of association of the time with the quantity.

3.3.1 Challenges

- Temporal expression identification and normalization is not simple because mentions are present in diverse surface forms. (e.g. "Dec. 2012", "last year"). Any system with a decent recall cannot rely on simple regular expression based matching for the task.

- Due to the absence of any structure, temporal association in text is challenging. For example in Figure 1.2, the quantity in snippet *t1* is separated from its relevant temporal expression by multiple tokens. Also, there are other temporal expressions in the snippet ("1960").

	Population in 2010
China	1,339,724,852
India	**1,182,105,000**
Australia	22,299,800
...	...

Figure 3.3: Sample Table

- The presence of some, albeit fuzzy, structure in web tables, makes the problem of temporal association slightly easier in tables when temporal information is present in the table. For example in Figure 3.3, the presence of temporal information in the same header as the table snippet, indicates strongly that the temporal mention is associated to the extracted quantity.

- However in the absence of temporal expression inside the table, we have to fall back to extraction and association of temporal expressions from context text found in the vicinity of the table. For example, in Figure 1.2, for the quantity extracted from table *t1* the relevant temporal expression is present in one of the many context texts.

Since this is an important component of our system, and since there is little prior work on how to accurately solve this task, we elaborate on how we solve these two steps in Chapter 4.

3.4 Response generation as a collective temporal model

The candidate table and text snippets, along with their relevance scores, possible number and unit extractions, and candidate time associations, are input to a collective model that we describe in Chapter 5. The output of the model is a response to be presented to the user as a time-varying distribution of the quantity.

Chapter 4

Temporal clues extraction and scoring

An important part of the QTQsystem, is the module to extract, normalize and associate temporal expressions to snippet quantities. Due to the challenges of the task, explained in Chapter 3, the task is highly non-trivial. In this chapter we present our method of extracting temporal clues and scoring them for both table and plain text snippets.

4.1 Extracting and Normalizing Temporal clues

The first step towards temporal association, is recognizing temporal expressions in text and tables, and extracting their correct value. Temporal expressions found can be broadly categorized into one of the three following types,

1. Explicit Temporal Expressions - Expressions of the type *Jan 30, 2011*. Normally such temporal expressions are extracted using regular expressions and normalization is trivial.

2. Implicit Temporal Expressions - Expressions like *Independence Day, 2010*. Such temporal expressions require world knowledge and context disambiguation to extract and normalize.

3. Relative Temporal Expressions - Temporal expressions like *tomorrow* or *last year* are relative temporal expressions as they can only be normalized relative to an anchor date.

Using just regular expressions, for temporal expression extraction, gives poor precision and recall. To give an example of the challenges in the task, consider the following few sentences.

- "John Doe, the movie star who born on the 9th of March, 1983, died yesterday."

  - Normalization of the value of *yesterday*

- "On 15th August of the year 1957, India became Independent"
  - *15th August of the year 1957* is one temporal expression

- "Independence Day"
  - Temporal expression whose normalized value depends on the context

Most temporal expression extraction and normalization systems are deterministic and rule based. HeidelTime ([Strötgen and Gertz, 2010]) is one such system, which is a part of the UIMA pipleline[1], and annotates and normalizes temporal expressions according to the TimeML standards[2]. In our experiments, we found HeidelTime gave the best qualitative results, when compared to SUTime ([Chang and Manning, 2012]) and GUTime ([Verhagen and Pustejovsky, 2012]). HeidelTime handles every temporal expression as a three tuple $< e_i, t_i, v_i >$, where e_i is the expression as it appears in text, t_i is the type of the temporal expression (one of *DATE, TIME, DURATION, SET*) and v_i is the normalized value. It uses hand-crafted rules for extraction and normalization of each type of temporal expressions. It has a database of world knowledge which is used to normalize implicit temporal expressions. To disambiguate relative temporal expressions, a rather straightforward approach of using the last mentioned concrete date as a reference date is used. A reference date can also be supplied for each document, and all relative temporal expressions in that document are then normalized using the given date.

4.2 Temporal Association

The next stage, is associating the extracted temporal expressions to snippet quantities. Consider the text *t1* from Figure 1.2 or the text "Population of India in 2010 had more than doubled from the figure in 1970 to reach 1.2 billion". The relevant temporal expression for a quantity, might not be the closest or even in the same sentence. Noisy text extracted from web pages, makes the problem harder. We now present our techniques for temporal association in text and tables.

[1] http://uima.apache.org/

[2] http://www.timeml.org

4.2.1 Text

We approach the problem of temporal association in text, as a supervised classification problem. For a text snippet s, containing quantity q and a list of temporal mentions $tm_1, .., tm_T$, we create a list of instances of value, temporal mention pairs. For each instance (v, t_t), we try to determine if the association holds, and use the association score. An important point to note here is, that multiple temporal mentions might represent the same time epoch. For example, in Figure 1.2 the text *t1* contains two temporal mentions for the time 2009. Since our further computations consider only distinct time epochs from each snippet, rather than temporal mentions, we use the best association score among all temporal mentions for a particular time epoch y as c_{sy}.

For supervision, we had text snippets annotated with the correct temporal association, for 23 queries. This gives us over 2500 training instances of value, time pairs. We train a logistic classifier over features extracted from the text between the annotated value and temporal expression, and a few tokens on either side. Broad types of features used are:

1. Features indicating sentence and segment boundaries
2. Preposition words indicating presence of temporal relation (e.g. "until", "since")
3. Features for common temporal patterns (e.g. "*quantity* in *time*")
4. Presence of temporal prepositions around the intermediate text
5. Normalized length of the intermediate text
6. Number of other temporal expressions in the intermediate text

4.2.2 Tables

Given that tables contain more structure than plain text, temporal association in tables is comparatively easier. If the there exists temporal information in the same row, or in the column header for the snippet cell, it is almost always correct. Therefore, our model works in the following steps:

- We look for temporal mentions in the column header, and return it with an association score of 1 if found
- Otherwise we look for temporal mentions in the same row, as the snippet cell. If found, the temporal mention is returned with an association score of 1

- If no temporal mentions are found in the column header or row, we search for temporal mentions in table contexts. Any temporal information in context texts, is likely to be associated to all information in the table. Therefore, we used a supervised model to judge the relevance of the temporal mention in context, to the query words. The model uses features similar to the temporal association model for free text.

Chapter 5

Overall Model

For each snippet s, in addition to the relevance score r_s, possible extractions of value characterized by t_{sj}, v_{sj}, g_{sj}, we have candidate temporal value Y_s and each $y \in Y_s$ is associates with a score c_{sy} that indicates the probability $\Pr(y|s)$ that time y is the valid time for the extracted value in the snippet. The $\Pr(y|s)$ values are obtained from the temporal association model.

Let $h_\tau(v|y)$ be a suitably trained regression function that represents the conditional distribution of value v at time y. We train this function by using v_{sj}, y, w_{sjy} obtained over different snippets s, their possible extractions j, candidate temporal associations y and weight w_{sjy} denoting the possibility of snippet s having value v_{sj} at time y.

5.1 Collective Inference Algorithm

The algorithm that we use for adjusting individual extraction scores by consensus is the collective inference algorithm, explained in Figure 5.1. The algorithm takes as input, all the snippets s and its relevance score r_s. With each snippet, we have the list of candidate extractions, for each unit tagged : $(t_{s1}, v_{s1}, g_{s1}), .., (t_{sj}, v_{sj}, g_{sj})$. Each snippet also contains the list of time epochs, identified in the snippet, with their association scores : Y_s, and scores c_{sy} for each $y \in Y_s$. For each value, time pair in snippet s, w_{sjy} is initialized.

The algorithm then iteratively updates the relevance scores r_s and weights w_{sjy} using a leave-one-out mechanism. In each iteration, for each snippet s, the model $h_\tau(v|y)$ is trained on all value, time pairs from all snippets except s. The probability of each candidate extraction j at each of the time epochs y in s, obtained from this retrained classifier, is used to update the weights w_{sjy} and r_s.

Inputs: $\Pr(t|q)$; for all snippets s, r_s and t_{sj}, v_{sj}, g_{sj} for all candidate extractions j, candidate time mentions Y_s, and scores c_{sy} for each $y \in Y_s$.

Evolving variables: $\check{r}_s, \check{w}_{sjy}$

initialize hidden variables $\check{w}_{sjy} \leftarrow g_{sj}\, r_s\, \Pr(t_{sj}|q)\, c_{sy}$

for iterations $i = 1, 2, \ldots$ **do**

  for each snippet s **do**

    let $h_\tau^{\setminus s}(\bullet)$ be a value distribution estimated from all snippets except s, using weights $\check{w}_{sjy}$

    for each candidate extraction j, time $y \in Y_s$ **do**

      $\phi_{sjy} \leftarrow g_{sj}\, r_s\, \Pr(t_{sj}|q)\, c_{sy} h_{t_{sj}}^{\setminus s}(v_{sj}|y)$ {consensus}

    end for

    $\phi_{s\perp} \leftarrow 1 - r_s$

    $D \leftarrow \phi_{s\perp} + \sum_{j,y} \phi_{sjy}$

    $\check{r}_s \leftarrow (1/D) \sum_{j,y} \phi_{sjy}$

    for each j, y **do**

      $\check{w}_{sjy} \leftarrow \phi_{sjy}/D$

    end for

  end for

end for

Figure 5.1: Collective inference pseudocode.

5.2 Temporal regression models

The input to these models is a sequence of values $\{v_1, \ldots, v_n\}$, time values $\{y_1, \ldots, y_n\}$, and a weight $\{w_1, \ldots, w_n\}$ indicating the importance of each time-value pair. Goal is to build a model $h(v|y)$ that can provide a probability distribution over the values v at each time y using the v, y, w series. Another desiderata from the model is efficient retraining when weight of a single instance is modified.

This can be modelled as a regression problem, and there are many options, including least square, spline models, kernel regression, and Gaussian Process (GP) regression. Since we want a probability distribution, and not just a point prediction at each t, we rule out pure time series models such as ARIMA models. Of the models that output a distribution, the GP approach appeared the most promising because of its non-parametric appeal.

We first provide background on GP regression. Then we show how we perform efficient retraining when we update weight of a single instance.

5.2.1 Gaussian Process Regression

The starting premise of GP regression is that the value at each time y follows a Gaussian distribution with mean μ_y and variance σ_y where both μ_y and σ_y are function of y. Also, if $\mathbf{v}_D = v_1, \ldots, v_n$ are values at n finite time points $\mathbf{y}_D = t_1, \ldots, t_n$, then they jointly follow a multivariate Gaussian distribution. The covariance between any two v_i, v_j is proportion to $K_t(y_i, y_j)$ a kernel that measures the "proximity" between i and j. This has the natural interpretation that values closeby in time are more correlated than far-off values. The covariance matrix K_D defining the joint Gaussian over the observed n points is thus a Gram matrix of the kernel where the i, j entry is $K_t(y_i, y_j)$ (we used a RBF kernel where $K_t(y_i, y_j) = e^{-d(y_i - y_j)^2}$ where d is a kernel width parameter. For the i, i entry, it is customary to add a noise parameter δ to capture the uncertainty of the observed value.

Using the GP framework, we can obtain a probability distribution at any unknown time value y. Any new time point y induces a new $n + 1$ dimensional Gaussian distribution defined at the n points and y. The covariance matrix is defined exactly as above. From this $n + 1$ dimensional Gaussian, we can obtain the distribution for any point y by conditioning on the remaining variables. We skip the details of the derivation [Mackay, 1998], and present the formula for the mean and variance of this conditional distribution $\Pr(v|y, D)$ as $\Pr(v|y, D) \sim N(\mu_y, \sigma_y)$ where

$$\mu_y = K_t(y, \mathbf{y_D}) K_t(\mathbf{y_D}, \mathbf{y_D})^{-1} \mathbf{v_D},\ \sigma_y^2 = K_t(y, y) - K_t(y, \mathbf{y_D}) K_t(\mathbf{y_D}, \mathbf{y_D})^{-1} K_t(y, \mathbf{y_D})^T \tag{5.1}$$

We can easily modify the above to include weighted points by multiplying $k_t(y_i, y_j)$ with $w_i w_j$, for $ij \in D$, multiplying $k_t(y, y_i)$ by y_i and multiple v_i with w_i.

Incremental retraining A computational challenge of the above model is that the estimation of the mean and variance at a new point requires the inversion of a $n \times n$ matrix. When the set of observations is fixed, we can precompute the inverse $K_t(\mathbf{y_D}, \mathbf{y_D})^{-}1$ in advance, and then the operations required during prediction are quadratic in n instead of cubic. In our case, however, the collective model, requires the leave one out probabilities which causes the set in D to change constantly. We cannot afford to recompute the inverse with every change in D. We exploit the

ideas presented in [Salmen et al., 2010] to efficiently update the inverse when the weights of only only point changes.

5.2.2 Conditional Kernel Density

The GP model performs well on clean training data when values at a time t are uni-modal and follow a Gaussian distribution. In our setting, the input values are candidate extractions from noisy web snippets with possibly incorrect weights, and the values are far from uni-modal. One of the possible reasons for a multi-modal nature of the distribution, might be the inherent ambiguity in the query. For example, many snippets retrieved for queries with the attribute "CO2 emissions", contain the value of emissions resulting from various fuel types as opposed to the "total" value for emission.

We designed the following method to estimate these densities 2D kernels.

$$\Pr(v|t) \propto \sum_{i=1}^{n} \frac{w_i e^{(t-t_i)^2/2\sigma^2}}{\sqrt{2\pi}\sigma_i} e^{(v-v_i)^2/2\sigma_i^2} \tag{5.2}$$

The above formula applies kernel density estimation at each point t where the weight of a value v_i is evaluated as a product of the initial weight w_i and a kernel that measures the distance between t and t_i. Alternately, it can be viewed as a 2-D kernel density along the time and value dimensions, with only diagonal co-variance terms. The variance σ^2 along the time dimension is a constant but along the value dimension it is allowed to change with the value. The reason for this difference is that values extracted from noisy web snippets tend to be at arbitrarily different scales, whereas the values for time fall within a narrow range because of the precision with which time (year, in our experiments) is extracted.

Chapter 6

Experiments

6.1 Workload

Corpora We use the web as our corpus, leveraging web search engines for retrieving relevant documents. For our query workload, we retrieved pages, created snippets and stored them. The extracted table store from web pages was augmented with the table store created by [Sarawagi and Chakrabarti, 2014]

Queries and ground truth Our experiments were performed on 268 queries. The query sources include

- **WorldBank** 172 queries on four attributes of countries: forest area, co2 emissions, land area, and population. Ground truth was extracted from World Bank data.worldbank.org
- **InfoGatherer** 96 queries across temporally varying attributes: revenue and profits for 31 large corporations and corporate tax rate for 34 countries.

6.2 Performance measures

For each query, the ground truth G is available as a value g_t at each time t at the granularity of a year. The year values present in G varies with query. Since, we cannot hope to find the exactly value match in the retrieved answers, following [Banerjee et al., 2009a, Sarawagi and Chakrabarti, 2014], we assume that the query specifies a multiplicative confidence band ϵ. I.e., if a true point value is g, the user would be satisfied with a value in $[(1-\epsilon)g, (1+\epsilon)g]$ (Our

results are with $\epsilon = 0.05$). Meanwhile, QTQ presents a distribution $h(v|t)$. The probability G in h is measured as

$$\sum_{t \in G} \int_{v=(1-\epsilon)g_t}^{(1+\epsilon)g_t} h(v)dv, \tag{6.1}$$

which is the total area matching a ground truth value band.

In order to relate to conventional IR measures of retrieval efficacy, we also report a soft version of mean average precision (MAP) measured as follows. Using the weights w_{sjy} output by the algorithm in Figure 5.1, we aggregate weights of "close" value,time pairs. Two value,time pairs are "close" when they match on time, and their values fall within an ϵ band of each other. We then generate a ranked list of time,value pairs $v_1, t_1, \ldots, v_K, t_K$ sorted on the aggregated weight. We measure MAP on this ranked list using $G = \{(g_t, t)\}$ as follows:

$$\frac{1}{|G|} \sum_{k=1}^{K} \delta\Big(g_{t_k} \in [(1-\epsilon)v_k, (1+\epsilon)v_k]\Big)$$

Data Sources We report results on three different data sources:

- **Text** - Uses only snippets extracted from text.
- **Tables** - Uses only tables snippets from the two table sources.
- **TextAndTables** - Uses snippets from both text and tables.

Our experiments show that using both text and table snippets outperforms using only text or tables.

	Collective	Independent
Tables	0.138525735	0.109232259
Text	0.113230513	0.069155002
TextAndTables	**0.144572313**	0.097289548

Table 6.1: Comparing data sources using Temporal Kernel Density Model - Average Probability

6.3 Collective inference

We show that collective inference outperforms the independent model. We show an improvement of 20% for MAP and 32% for average probability of ground data. This reaffirms our

	Collective	Independent
Tables	0.0991265	0.084753233
Text	0.033009744	0.029138897
TextAndTables	**0.119647444**	0.099371035

Table 6.2: Comparing data sources using Temporal Kernel Density Model - MAP

belief that individual extractions from noisy web data are highly unreliable, and early hardening of such extractions hurts performance. A consensus on the other hand, is successful in weeding out noisy extractions, and increasing our belief in correct ones.

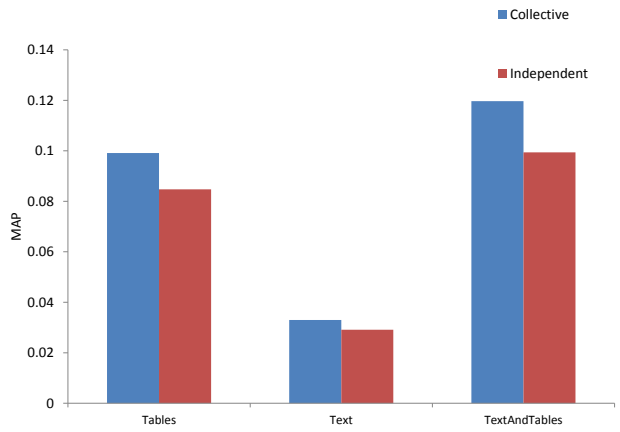

Figure 6.1: Comparing collective model with independent - MAP

6.4 Choice of Temporal model

We find in our experiments, that the Temporal Kernel Density model, beats the Gaussian Processes model for all settings of data sources and extraction methods (collective and independent) on both MAP and average probability. As observed earlier, this is due to the noisy nature of extractions from the web.

6.5 Comparing Performance on Recent Extractions

We compare the performance of collective method on gold data from 2004 to 2013 against performance on all ground data. We see that from our data that a large number of extractions

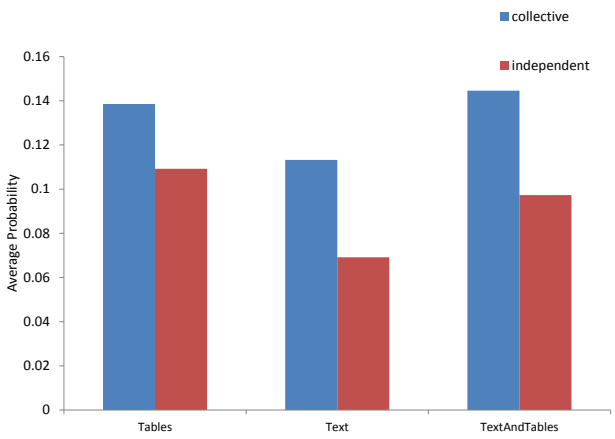

Figure 6.2: Comparing collective model with independent - Average Probability over all queries and gold time epochs

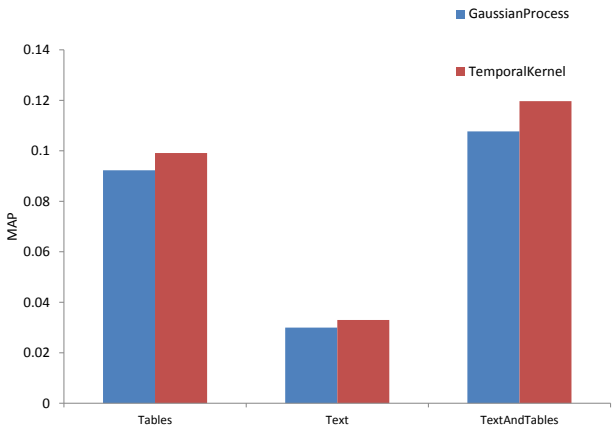

Figure 6.3: Comparing Temporal KD model with Gaussian Processes - MAP

are for recent years. The improved performance shows that the presence of more candidate extractions (albeit slightly noisy), helps the overall performance.

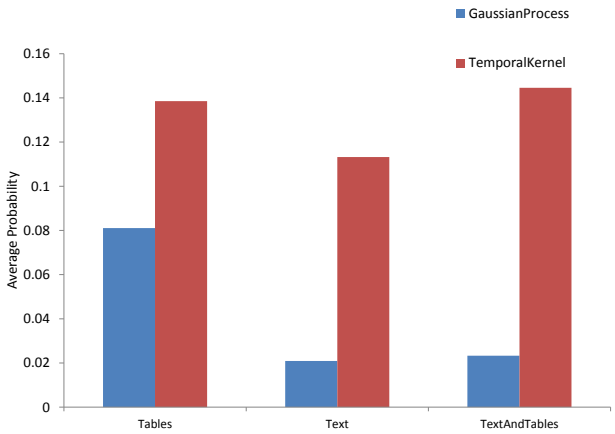

Figure 6.4: Comparing Temporal KD model with Gaussian Processes - Average Probability over all queries and gold time epochs

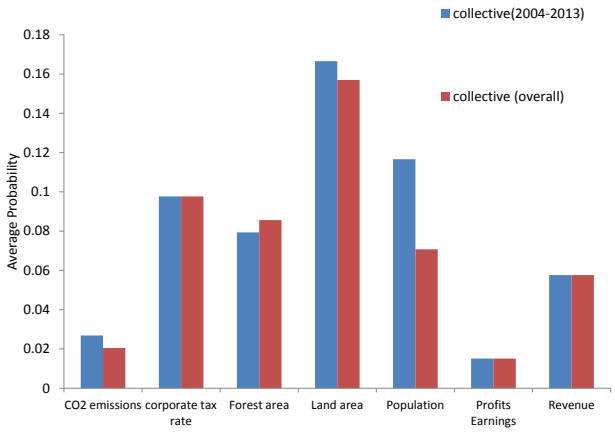

Figure 6.5: Comparing performance on recent ground data against performance on all data

Chapter 7

Discussion and Future Work

The two main contributions we make with QTQare -

- Answering fluent quantity queries
- Combining extractions from both free text and tables for answering queries

As we see from the results, our method of combining extractions from text and tables, achieves marked improvement over using only text or tables. The collective method of assigning extraction scores, by consensus, which was shown to work well for temporally agnostic quantity answering system in [Sarawagi and Chakrabarti, 2014] also performs better with the modification to handle temporally varying quantities. We will now list some shortcomings and areas for future work.

Temporal Model We show that the Temporal Kernel Density model outperforms the Gaussian Processes model. This is mainly because for most temporal quantities the distribution is not unimodal. However, allowing our distribution to be multimodal makes it slightly less robust to weeding out noisy extractions in the collective model. Also, the Temporal KD model is not as adept at interpolation for unobserved time values. Therefore, we think a better temporal model more robust would help improve performance.

Better Text Snippet Relevance Scores We observe that the relevance scores obtained from the rank SVM model are not very well calibrated. Creating more stringent filters is not a decent option for us, as we need as many candidate values as possible, to observe temporal variations in the quantity. The poor scores lead to problems downstream where many noisy values are

scored higher than some correct values, and compound errors in the collective model. We also observe that the main goal in the QCQ paper [Banerjee et al., 2009b] was to create better ranking functions for tight value bands and the scores from the individual snippet ranking model were just one of the many features they used for ranking bands of values. We believe we can improve snippet ranking on QCQ with some simple additional features.

Distant Supervision One of the interesting future directions for this work, is to train the temporal association model using *Distant Supervision*. Distant Supervision is a framework, where a database of known facts or ground truth data (which is easily available), can be used to annotate data for use by a *Supervised model*. The advantage of such an approach is, that it saves human effort required for manual annotation. For our problem, we could use the ground truth data which is the gold list of time, value pairs, to train a probabilistic model (a *labeller*), similar to the regression model in the collective method. For annotating a candidate snippet (v, t), we could output the probability of the v being true at time t from our *labeller*. This probability could be used as a soft label instead of hard true, false manual labels.

Dependency Parse A dependency parse of text is used to represent the relation between various parts of the text. For example, for the sentence "My dog also likes eating sausage.", the following dependencies are output by the Stanford Parser[1].

poss(dog-2, My-1)
nsubj(likes-4, dog-2)
advmod(likes-4, also-3)
root(ROOT-0, likes-4)
xcomp(likes-4, eating-5)
dobj(eating-5, sausage-6)

Using such dependencies as features for ranking snippets, could drastically reduce noisy snippets. However, creating a dependency parse is time consuming, and its value on problems with web data, is uncertain.

[1] nlp.stanford.edu:8080/parser/

Bibliography

[Bakalov et al., 2011] Bakalov, A., Fuxman, A., Talukdar, P. P., and Chakrabarti, S. (2011). SCAD: collective discovery of attribute values. In *WWW Conference*, pages 447–456.

[Banerjee et al., 2009a] Banerjee, S., Chakrabarti, S., and Ramakrishnan, G. (2009a). Learning to rank for quantity consensus queries. In *SIGIR Conference*.

[Banerjee et al., 2009b] Banerjee, S., Chakrabarti, S., and Ramakrishnan, G. (2009b). Learning to rank for quantity consensus queries. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 243–250, New York, NY, USA. ACM.

[Chang and Manning, 2012] Chang, A. X. and Manning, C. (2012). Sutime: A library for recognizing and normalizing time expressions. In Chair), N. C. C., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

[Joachims, 2002] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *SIGKDD Conference*, pages 133–142. ACM.

[Mackay, 1998] Mackay, D. J. (1998). Introduction to gaussian processes.

[McClosky and Manning, 2012] McClosky, D. and Manning, C. D. (2012). Learning constraints for consistent timeline extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 873–882, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Moriceau, 2006] Moriceau, V. (2006). Numerical data integration for cooperative question-answering. In *EACL Workshop on Knowledge and Reasoning for Language Processing*, pages 42–49.

[Pimplikar and Sarawagi, 2012] Pimplikar, R. and Sarawagi, S. (2012). Answering table queries on the web using column keywords. In *Proc. of the 38th Int'l Conference on Very Large Databases (VLDB)*.

[Salmen et al., 2010] Salmen, J., Schlipsing, M., and Igel, C. (2010). Efficient update of the covariance matrix inverse in iterated linear discriminant analysis. *Pattern Recogn. Lett.*, 31(13):1903–1907.

[Sarawagi and Chakrabarti, 2014] Sarawagi, S. and Chakrabarti, S. (2014). Open-domain quantity queries on web tables: Annotation, response, and consensus models. In *ACM SIGKDD*.

[Strötgen and Gertz, 2010] Strötgen, J. and Gertz, M. (2010). Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324, Uppsala, Sweden. Association for Computational Linguistics.

[Verhagen and Pustejovsky, 2012] Verhagen, M. and Pustejovsky, J. (2012). The tarsqi toolkit. In Chair), N. C. C., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

[Wu and Marian, 2007] Wu, M. and Marian, A. (2007). Corroborating answers from multiple web sources. In *WebDB: Tenth International Workshop on the Web and Databases*.

[Zhang and Chakrabarti, 2013] Zhang, M. and Chakrabarti, K. (2013). Infogather+: Semantic matching and annotation of numeric and time-varying attributes in web tables. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 145–156, New York, NY, USA. ACM.