Automatically Generating Interesting Facts from Wikipedia Tables

Flip Korn
Google Research NYC
flip@google.com

Xuezhi Wang
Google Research NYC
xuezhiw@google.com

You Wu
Google Research NYC
wuyou@google.com

Cong Yu
Google Research NYC
congyu@google.com

ABSTRACT

Modern search engines provide contextual information surrounding query entities beyond *ten blue links* in the form of information cards. Among the various attributes displayed about entities there has been recent interest in providing fun facts. Obtaining such trivia at a large scale is, however, non-trivial: hiring professional content creators is expensive and extracting statements from the Web is prone to uninteresting, out-of-context and/or unreliable facts.

In this paper we show how fun facts can be mined from *superlative tables* in Wikipedia, whose rows are ranked according to some statistics, to provide a large volume of reliable and interesting content. We employ a template-based approach to semi-automatically generate natural language statements as fun facts. We show how to bootstrap and streamline the process for faster and cheaper task completion. However, the content contained in these tables is dynamic. Therefore, we address the problem of automatically maintaining the pairing of templates to tables as the tables are updated over time. Fun facts produced by our work is now part of Google's production search results.

ACM Reference Format:
Flip Korn, Xuezhi Wang, You Wu, and Cong Yu. 2019. Automatically Generating Interesting Facts from Wikipedia Tables. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3299869.3314043

SIGMOD '19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5643-5/19/06.
https://doi.org/10.1145/3299869.3314043

1 INTRODUCTION

Displaying interesting trivia, or *fun facts*, is a strategy increasingly employed by search apps to increase user engagement. One way to surface these in search results is through information cards such as Google Knowledge Panels, Bing Satori, or Yahoo! Knowledge, for entity-seeking queries; see Figure 1 for an example.

Efforts to mine fun facts from the Web, rather than hiring professionals to curate material, are motivated not only by cost and scalability—the throughput reported in [17] was 50 trivia statements per day—but also reliability (facts must be sourced from accurate material, self-contained and phrased precisely) and freshness (facts can easily become stale). Hence, there have been attempts to automate this process. An early attempt was based on selecting (unstructured) text in IMDb pages using domain-specific training data [17]. Unfortunately, text extraction is prone to pulling statements out of context, as demonstrated in [23], due to challenging issues like multi-sentential co-reference resolution. Wikipedia has thus served as a rich source for *generating* fun facts due to its associated structured data, such as triples extracted from entity pages [8, 18, 19] and category lists at the bottom of pages [23]. As (non-comparative) examples, the former could generate a statement like, *The movie Frozen was produced by Peter Del Vecho* from the triple $(Frozen, produced\_by, Peter\_Del\_Vecho)$ whereas the latter could generate *Frozen is in the category of "feminist films".*

Until now, and complementary to these existing approaches, relational tables [3] on Wikipedia pages have been an untapped resource. Yet, unlike these previous approaches, tables are able to capture the uniqueness and importance of an entity *in relation to other entities.* An example, again for the movie entity *Frozen*, is the statement *Frozen is the highest-grossing animated film of all time.* Generating fun facts from relational tables leads to an abundance of interesting trivia statements without the same pitfalls that these previous approaches face (extracting sentences out of context, choosing

Figure 1: Example facticle at the bottom of the Google Knowledge Panel for the movie *Frozen*

unexceptional or esoteric facts, etc.) and, as we demonstrate in Section 6, users prefer them.

In this paper, we present details from our approach for generating natural language trivia statements from relational tables in Wikipedia. Given the popularity of ranked lists ("listicles") and entity comparisons (e.g., [22]), we focus on *superlative tables* as a natural source of interesting facts; we call the resulting statements *facticles* (for facts from tables). Our contributions are as follows:

- We show how to identify from Wikipedia a large number of relational tables, and attributes within each table, that are an excellent source for generating fun facts.
- We propose a templated approach that, when instantiated using table data, automatically generates many fun facts from a single table. This approach is reusable for any update to the table data (i.e., non-schematic), such as row reorderings due to changes in measure values, and can thus stay up-to-date and avoid making stale statements.
- We propose two general classes of templates—*rank-ordered* and *distributional*—that lead to interesting sentences when instantiated with table data.
- We give a semi-automated method for turning structured facts from tables into natural language templates. Starting from a small collection of curated templates, each corresponding to a table, we developed a language model that accurately, though not perfectly, derives templates for new and unseen tables. Modifying these templates into grammatically correct natural language requires only lightweight work and, therefore, can be done cheaply and efficiently (e.g., via a crowdsourcing system like AMT).
- We propose a method for dynamically maintaining (table, template) pairings over time that gracefully handles table schema changes like column header renamings and column reorderings.
- We present experiments demonstrating that the fun facts generated by our approach are interesting to users and preferable to those by existing approaches; the manual effort to inspect and correct our templates is low; and our ability to correctly pair tables and templates over time is good, resulting in very few false-positives and false-negatives.
- All those characteristics of the system enabled *facticles* to be launched into the production search results of Google (as depicted in Figure 1) and serves tens of millions of users per day.

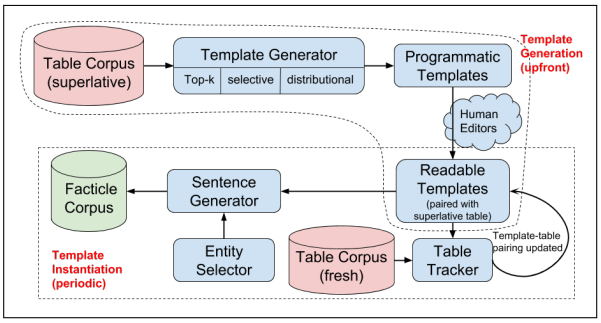

Figure 2: Pipeline for generating facticles

Figure 2 depicts the pipeline once training data has been collected (see Section 4.1) and a template generation model has been trained (see Section 4.2). Given an input table corpus, the model initially generates candidate templates based on the views described in Section 3, which are then manually edited upfront. The table-template correspondences are maintained dynamically over time (see Section 5) and periodically instantiated based on the tables to generate English sentences. At serving time, the (pre-computed) sentences are indexed by entity and surfaced on query. Section 6 provides experimental evidence of the interestingness of facticles, the accuracy of our templated approach and the effectiveness of table tracking. Section 2 summarizes related work and Section 7 gives our conclusions.

2 RELATED WORK

Approaches for mining trivia fall into two general categories: those that extract sentences from unstructured text corpora and those that generate sentences from structured data. Below we discuss the corpora employed and methods used for extracting or generating statements by these approaches.

Among unstructured approaches, the WTM algorithm from [17] employs a supervised learning model (Rank SVM), using crowd-voted IMDB trivia section sentences as training data, to find the top-k interesting sentences from a Wikipedia movie pages based on linguistic features. A major pitfall of this approach is that sentences may not be self-contained and often do not make sense when pulled out of context due to co-reference, etc. Among structured approaches, [23] exploits hypernym categories located at the bottom of Wikipedia pages and selects based on "surprise" (the entity X is an outlier for the category) and "cohesiveness" (the category Y contains a limited variety of entities) to generate statements of the form "X is a member of Y".

Natural Language Generation (NLG) is the problem of transforming structured data, such as RDF triples, into human-comprehensible text [15], as opposed to Information Extraction (IE), which extracts structured relations from natural language (cf., [20]). Within this thread, [18] and [8] employ recurrent neural networks to transduce a Freebase triple into natural language questions for which the triple ("fact") would provide an answer to its corresponding generated question. The resulting text, however, are intended to be factoids, not trivia, sometimes with multiple correct answers (e.g., "What's the name of a place within Illinois?") and are not specifically generated based on interestingness. [11] uses a neural approach to generate biographical text from name-value pairs of a single entity's Wikipedia Infobox. [24] generates descriptions of entity relationships, for a given relationship instance, using Freebase and a set of relationship instances combined with their descriptions based on Wikipedia articles; for example, `starsInFilm(Brad Pitt, Troy)` could be used to generate the sentence "Brad Pitt appeared in the American epic adventure film Troy." It does this by first creating sentence templates for a particular relationship and then generates each textual description by selecting the best template and filling it with appropriate entities. [19] shows how to transform questions into multiple-choice form and how to select distractors (choices besides the correct answer) based on level of difficulty.

To the best of our knowledge, the only prior work for generating trivia from tables is [12], which randomly chooses relational operators for iterative composition, and randomly chooses attributes and values from the table for instantiation, ordering candidates based on selectivity as well as domain-specific heuristics (e.g., *Best Picture* is a good attribute for a movie table); the result is a relational expression instead of a natural sentence. Unfortunately, no details about candidate scoring or experimental results were reported in [12].

Finally, we mention a few related problems with different goals. In *table answering*, a keyword query is posed and the goal is to extract an answer, from a corpus of tables, returned as a scalar/tuple or a set/table, depending on how many answers there are to the question and whether auxilliary attributes should be returned [2, 14, 16]. *Data verbalization* and *narrative writing* have the goal of providing short textual descriptions or summaries of a dataset, such as reporting important events of a sports game or providing an overall outcome using box-scores [1, 25]. *Textual explanation* aims to translate the query into a natural language narrative given a specified structured (SQL) query [9, 10, 21]. *Reading comprehension* question generation has the goal of generating questions to test the reader's understanding of a given text passage such as a sentence [5].

Table 1: List of Tallest Buildings in the World

Rank	Building	Country	Height(m)	Height(ft)	Floors
1	Burj Khalifa	UAE	828	2717	163
2	ShanghaiTower	China	632	2073	128
3	Abraj Al-Bait	Saudi	601	1971	120
4	Ping An	China	599	1965	115
5	Lotte World	Korea	554.5	1819	123

3 DEFINITIONS AND VIEW CLASSES

In this section, we propose two general template view classes that, when instantiated with a specific entity, can generate interesting facts about the entity in relation to others. The *rank-ordered* view class describes how exceptional an entity is compared to other entities in a given set, with respect to some given ordered attribute, and the *distributional* view class describes how exclusive an entity is compared to the other entities with respect to membership of some given unordered attribute value.

For illustration purposes, Table 1 shows an example superlative table with schema `(Rank, Building, Country, Height(m), Height(ft), Floors)`.[1] Each row of the table can be thought of as corresponding to a single entity (in this case, a building) with the columns containing attribute values for each respective entity. We make use of several important columns: *member column*, *measure column*, and *category column*. Member column gives the names of the per-row entities in the table and can be thought of as functionally determining most of the other attribute values. Measure column provides a numeric value that is important to what the table summarizes. Finally, category column partitions the entities into groups. For example, in Table 1, the member column would be `Building`, the measure column could be either `Height(m)` — height measured in meters — or `Height(ft)` — height measured in feet, and a category column would be `Country`.

We assume a table has exactly one member column, since that is the case for most tables of interest, and identify it using signals like the presence of anchored or inferred entities, the

[1] https://en.wikipedia.org/wiki/List_of_tallest_buildings#Tallest_buildings_in_the_world_(350_m+)

non-existence of temporal or numeric values, and the column position within table, as well as techniques from [2, 4].

3.1 Preliminaries

A *superlative table* contains one column pertaining to a *member attribute* and at least one column pertaining to a *measure attribute*, and can be rank-ordered according to values in the measure attribute.[2] An *entity class name* is a word or phrase describing a set of entities. For example, "skyscrapers" and "buildings in the world" are both valid class names for the entities in the member column in Table 1. A *superlative measure phrase*, or *superlative* for short, is the adjective phrase describing the extrema of values in the measure column. For example, a superlative describing the building with maximum height in Table 1 is the word "tallest". A *measurement unit* is a term used to describe the units along which the measure attribute values are ranked. For example, the measurement unit for attribute `Height(m)` is "meters". A *categorical attribute* is a (typically unordered) column from the table that is not a member or measure attribute and can be used to partition the rows by the distinct values into groups. For example, `Country` is a categorical attribute in Table 1.

3.2 View Definitions

Definition 1. *A* rank-ordered view *takes a tuple* $\langle member, measure \rangle$*, optional selectors* $\{\langle selector_i, \sigma_i \rangle\}$*, and parameter* k*, and applies the query*

```
SELECT member, measure FROM T
[WHERE selector_1 = σ_1 AND ... AND selector_ℓ = σ_ℓ]
ORDER BY measure [DESC] LIMIT k
```

to some table T, with descending marker optionally added based on the detected order of values for the measure column. □

For example, a simple top-k view from Table 1 is `SELECT Building, Height FROM T ORDER BY Height DESC LIMIT 5`, for $k = 5$, since the member attribute is `Building` and the measure attribute is `Height`. In the following section we discuss how this view can be used to generate the template

`[$entity] is the [$rank] [$superlative] [$entity_class]`

where, for this table, the superlative is "tallest" and the entity class could be "buildings in the world"; hence this template can then be instantiated using a single row from the table, producing statements such as "*Shanghai Tower* is the 2*nd* tallest building in the world." from the second row.

As another example, using $\langle$`Country`, *China*$\rangle$ as a selector and $k = 5$, we can generate a selective top-k view as `SELECT Building, Height(m) from T WHERE Country = 'China' ORDER BY Height(m) DESC LIMIT 5`. This view can be used to generate the template

`[$entity] is the [$rank] [$superlative] [$entity_class] with Country` *China*

which can then be instantiated using the category value *China* as "*Shanghai Tower* is the tallest building with country *China*."

While the actual value of k is subjective, in production we use $k = 10$ given the prevalence of Top-10 lists.

Definition 2. *A* distributional view *takes a tuple* $\langle member, category \rangle$*, with optional selectors* $\{\langle selector_i, \sigma_i \rangle\}$*, and defines the view V as*

```
SELECT category, COUNT(*) as count FROM T
[WHERE selector_1 = σ_1 AND ... AND selector_ℓ = σ_ℓ]
GROUP BY category
```

and then applies the query

```
SELECT T.category, T.member, V.count
FROM T JOIN V ON T.category=V.category
```
□

For example, using `Country` as the grouping category attribute, the distributional view on Table 1 would be based on `SELECT Building, COUNT(*) FROM T GROUP BY Country`. In the following section we discuss how this view can be used to generate the template

`[$entity] is one of [$count] [$superlative] [$entity_class] with [$category_name] [$category_value]`[3]

which could then be instantiated using the category value *China*, the `[$count]` of tuples selected using this group and the number `[$N]` of tuples from Table 1, as "2 of the 5 tallest buildings in the world have Country *China*, including *Shanghai Tower*."

While choosing a threshold for reporting interesting distributional observations is subjective, we make use of entropy as follows: for each category attribute from the table T, we compute its normalized entropy $-\frac{1}{\log N}\sum_i (c_i/N)\log(c_i/N)$, where c_i is the count of each category value i, and $N = \sum_i c_i$. Intuitively, a category attribute with lower entropy (non-uniform) is preferred. Based on trial-and-error, we chose a category attribute for the distributional view in production only when its normalized entropy was < 0.75.

3.3 Extension to View Combinations

If there are multiple measure columns including $measure_1$ and $measure_2$ in the same table, such as `Height` and `Floors` in Table 1, then rank-ordered views on $\langle member, measure_1 \rangle$ and $\langle member, measure_2 \rangle$ can be combined using the phrase

[2] In practice, we may not require a strict monotonic ordering of values along the measure attribute column but rather that, say, at least the first k values are monotonic, or that no more than k inversions can occur, to compensate for data quality issues in the table row ordering.

[3] An alternative phrasing that we use is
`[$count] of the [$N] [$superlative] [$entity_class] have [$category_name] [$category_value], including [$entity]`

"as well as", for example, "*Lotte World* is the 3rd tallest building by height as well as the 5th tallest building by number of floors." Combining two distributional views we can get "*Ping An* is one of the 3 tallest buildings in China as well as one of the 2 tallest buildings in Shanghai." Measure columns in different but compatible tables (over the same set of entities) can similarly be combined. For example, using the Wikipedia "List of Tallest Freestanding Structures" table whose schema includes attributes `Name`, `Pinnacle height`, `Country` and `City`, in combination with Table 1, we can obtain "*Shanghai Tower* is the 2nd tallest building as well as the 3rd tallest freestanding structure." We employ embedding similarity from corresponding components in the clauses to avoid incoherence between clauses, as described in Section 4.2.

4 TEMPLATE GENERATION

In Section 3, we introduced views that can be used to generate natural language sentences. In this section, we discuss the intermediate step of how to transform views into *templates* which can then be instantiated using data from the table to generate human-readable sentences, i.e., *facticles*.

4.1 Table selection

The first step in generating facticles is to find high-quality and interesting tables on the Web. While the top-k view is geared towards superlative tables, the distributional view in principle could apply to any distribution of category values. Identifying interesting columns for an arbitrary distributional view, however, is challenging; ones from superlative tables indicate potential influence or correlation on the superlative measure and hence are likely to be interesting. Creating superlative views on non-superlative tables is one direction we considered but eventually discarded for production—it is confusing for a user to see a facticle generated based on a table that cannot be found from the original source.

We started from Wikipedia's "Lists of Superlatives" category page[8], which organizes pages into a hierarchy, and selected roughly 350 of the most popular (leaf-level) pages by PageRank score. We processed all tables from these pages to identify member and measure columns and generated simple rule-based templates from them, of the raw form described in the previous section, by using the title as a stand in for the superlative and entity class. These templates were examined by multiple highly-skilled editors that we hired, who used Google Search throughout to understand the tables better and look up terms and then collaboratively refined them for readability. Most templates needed some modification and some needed a significant amount, with the per-template editing time ranging from about 30 seconds to ten minutes. In total, the task took roughly 80 person hours. These curated templates, and the tables corresponding to them, were then used as training data.

4.2 Natural Language Template Generation

Given a view, we generate templates using a language model that takes table metadata and tries to predict various template components from the view. For example, in Table 1, the top-k view

```
[$entity] is the [$rank] [$superlative] [$entity_class]
```

translates to the template:

```
[$entity] is the [$rank] tallest building
```[9]

where `[$entity]` and `[$rank]` are variables that are bound to the member value and sequence number of a given row, respectively. The components we need here include the entity class name "buildings", which is pertinent to the member column, and the superlative word "tallest", which is pertinent to the measure column and corresponds to the view clause `ORDER BY Height DESC`.

In addition, for any selection (or grouping, in the case of distributional views), we also generate a *phrasal verb* component corresponding to the clause `WHERE` $selector_i$ `=` σ_i. Again using table 1 as an example, the selective top-k view `[$entity] is the [$rank] tallest building` **`with Country [$Country]`** translates to the following template where the selection predicate is replaced with a more natural version:

```
[$entity] is the [$rank] tallest building
that is in [$Country]
```

where the relationship between the entity classes "buildings" and "countries" is described using the phrasal verb "is in", corresponding to the clause `WHERE Country = [$Country]`; for the selective view, but not for the distributional view, we further prepend the word "that". The entity classes are also used for conjugating the verb, by making use of information from the parser about singular vs plural, current vs past tense, etc., so that, for example, a single *building* would use the phrasal verb "is in" whereas multiple *buildings* would use "are in". Note the selection clause translation easily extends to distributional views, e.g., the distributional

[4] https://en.wikipedia.org/wiki/List_of_best-selling_Christmas_singles_in_the_United_States#Best-selling_Christmas_singles

[5] wikipedia.org/wiki/List_of_heaviest_land_mammals#Heaviest_land_mammals

[6] https://en.wikipedia.org/wiki/Fourteener#Fourteeners

[7] https://en.wikipedia.org/wiki/List_of_cities_and_towns_in_Iceland#Cities

[8] https://en.wikipedia.org/wiki/Category:Lists_of_superlatives

[9] In addition, we generate a suffix containing the measurement value along with its unit of measurement, e.g., `([$measure_value] copies of work sold)`.

Table 2: Examples of superlative phrases that are extracted from the title or inferred from metadata

Title	Superlative Phrase	Extracted or Inferred
Best-selling Christmas singles in the United States[4]	best-selling	Extracted
Heaviest land mammals[5]	heaviest	Extracted
Fourteeners of the United States[6]	highest	Inferred
Cities and towns in Iceland[7]	most populous	Inferred

view `[$count] of the [$N] tallest buildings` **`have Country [$Country]`** would translate to:

`[$count] of the [$N] tallest buildings are in [$Country]`

For each view class, we infer the various components needed by using features extracted from the table including titles, column headers, cell contents and surrounding text. In what follows we describe the process by which we generate these components.

4.2.1 Superlative Component. Table 2 gives examples of the correctly identified superlative phrases for each corresponding table. In cases where the superlative can be extracted directly from the title, it can usually be identified using the SyntaxNet parser[10] with Part-of-Speech tags for superlative adjectives (*JJS*) and adverbs (*RBS*). In cases where the title does not contain a superlative, we need an inference model for the superlative phrases. We analyzed a set of 1.1K tables and found that 53.3% of the tables require inference for the superlative phrase.

For example, the Wikipedia table "List of cities and towns in Iceland" is ranked by population and without considering the measure attribute we would only be able to say that the entities in the table are cities or towns in Iceland, which is incomplete and not particularly interesting. Instead, we make use of table metadata including the title, cell contents and column header name of the measure attribute whose values determine the ranking (which in this case is "2016 population"), to derive the superlative phrase "most populous".

Inference Model. We use both syntactic features and semantic features to build the inference model:

- Superlative phrase extracted from the table title. This feature is the most indicative but is not available for most of the tables;
- Root word identified from the title using the SyntaxNet parser, which indicates the main topic described in the table (e.g., the root word is "fourteeners" for the table title "Fourteeners of the United States");
- Bag-of-words vector for the member attribute column name (e.g., "Mountain Peak" for the fourteeners table);
- Bag-of-words vector for the measure attribute column name (e.g., "Elevation" for the fourteeners table);
- Bag-of-words vector for the table title;
- Hypernyms corresponding to entities identified in the member attribute column values after performing entity linking on the text (e.g., the hypernym is */collection/mountains* after performing entity linking on cell contents from the member attribute column: "Denali", "Mount Saint Elias", etc.);
- Embedding features obtained from table metadata (member attribute column name, measure attribute column name, title, surrounding text, etc.) after using a parser to segment the text into words and then mapping to word2vec embeddings trained on the Google News corpus[11]. Sentence-level features were obtained from a weighted sum over the word embeddings using corpus-based tf-idf weights. The embedding features provide better connections between semantically similar topics where the texts do not overlap, e.g., "fourteeners" and "peaks", "height" and "elevation".

Note that it is possible that some of the features described above are non-descriptive or noisy. For example, in some cases the member column name is simply `Name` or `Title` for many different topics. Hence, it is important to combine all these features for a better prediction.

We modeled the prediction as a multi-class classification problem, where the label classes are all the possible superlative phrases in the templates. The model yields a probability estimate along with the predicted superlative phrase; we tried simply taking the phrase with highest probability and that worked reasonably well though reasoning over multiple candidate superlatives may be beneficial in some cases.

For the extension to view combinations, we employed techniques from above to avoid incoherence between clauses. Each clause is mapped to a pretrained word2vec embedding weighted by corpus-based tf-idf scores and any paired clauses with cosine similarity below 0.7 were pruned.

4.2.2 Phrasal Verb Component. For the selective and distributional views, we also need to predict a phrasal verb

[10] https://github.com/tensorflow/models/tree/master/research/syntaxnet

[11] https://code.google.com/archive/p/word2vec/

[12] https://en.wikipedia.org/wiki/List_of_best-selling_books

[13] https://en.wikipedia.org/wiki/List_of_best-selling_Xbox_360_video_games#

Table 3: Examples of phrasal verb components inferred from tables

Title	Phrasal Verb	Category Column Name
Best-selling fiction authors	are	Citizenship
Best-selling fiction authors	wrote their works in	Original language
Best-selling books[12]	are written in	Original language
Best-selling Xbox 360 video games [13]	are published by	Publisher
Best-selling Xbox 360 video games	are	Genre

describing the relationship between the member and category columns. Table 3 gives some examples of phrasal verbs for various tables. Note the sensitivity of the phrasing of the relationship to the entity classes. For example, for the table of best-selling *books* rather than authors, no longer does the phrase "wrote their works in" make sense for the same category attribute `Original_language`; this would need to be replaced with "are written in".

Inference Model. Similar to the superlative prediction model, for phrasal verb prediction we also utilize features like column names, hypernyms and text embeddings, except that the features are extracted from both sides of the phrasal verb, specifically, the entity classes of the member attribute column and category attribute column, respectively. We used a multi-class model with the classes corresponding to phrasal verbs with the following types of features.

- Text features for the main entity class, or topic, of the table, including root word (identified from the title using the SyntaxNet Parser), bag-of-words vector for the member column name, and hypernyms corresponding to entities identified in the member column values;
- Text features for the category attribute, including bag-of-words vector for the category column name, and hypernyms corresponding to the entities identified in the category column values.

We include the text embeddings as well for the member column name and category column name to alleviate the data sparsity problem from bag-of-words representations.

5 DYNAMIC MAINTENANCE

Pairing templates with tables is necessary for being able to instantiate them into facticles. Moreover, pairing with the most recent version of each table is crucial for keeping facticles up to date. For many Wikipedia tables with open edit access, especially those that are superlative tables, the table data is likely to exhibit changes over time: measure values get updated, rows are inserted or deleted. As a result of this, (possibly many) rows get reordered. There are also occasionally table schema changes: columns get renamed, reordered, inserted and deleted. Rarely, there are even more complicated changes, like columns being split or merged, multiple tables being merged into one or a large table being split into multiple small ones. Such changes are important to detect not only for table tracking; in some cases, they require the template to be revised.

Our approach is to re-instantiate templates for any updated tables. Unfortunately, tables in Wikipedia do not have unique identifiers that would enable tracking them across time in the presence of updates. Note that while tables rarely move across pages (excepting for canonical URL renames), there is still a significant challenge in tracking tables from the same page since pages often contain multiple tables; therefore, page URL is not a viable identifier. We considered trying to make use of Wikipedia revision history logs for this but changes between two Wikipedia page snapshots must be inferred using a differencing algorithm[14] and no fine-grained provenance is kept. Table sequence number is not of much help either: the *n*th table on a page at one epoch frequently does not correspond to the *n*th table on the same page at a different epoch, as tables get inserted, deleted and moved around within the page. The table schema is also unreliable for maintaining correspondence between tables across epochs since multiple tables on the same page may have the same schema and a minor schema change could make two versions of the same table look distinct.

All these make tracking tables very challenging. Fortunately, it is rare that changes between two consecutive epochs of the same table (e.g., within a few days of each other) comprise a large fraction of the table. In our experiments, while on average 10% of the tables have at least one cell value changed from one epoch to another, only 0.23% of those changes exhibited a similarity below 0.5 based on the function we define below. Thus, in practice this does not require solving the notoriously difficult schema mapping problem.

5.1 Table Tracking

Our goal is to find the freshest version of each given table over time. As a proxy, we do this by dynamically maintaining each table's semantic counterpart and employ approximate matching techniques. Specifically, we propose a simple schema-agnostic similarity function between two tables that is invariant to row and column ordering but assumes that

[14] https://en.wikipedia.org/wiki/User:Cacycle/wikEdDiff

most of the cell values remain the same between two consecutive epochs. To allow for row and column reorderings, we define similarity $sim(S, T)$, between source and target tables S and T, using multiset intersection of their respective cell values. However, naively "flattening" all data cells of a table into a multiset allows cells from arbitrary rows and columns of tables to be equated, thus conflating semantic types. Therefore, we place limits on which cell values can be compared using two mechanisms that we describe as follows.

First, we restrict all cell values along any one column of the source table to be compared against only cell values from a single column in the target table. Then we find a mapping between columns in the source and target tables that maximizes overlap. Source and target columns are modeled as nodes in a complete bipartite graph, with edge weights corresponding to multiset intersection cardinalities; an optimal assignment can be solved efficiently using the Hungarian algorithm [7, 13].

Second, we *anchor* each non-member cell value c to the associated member column value m from the same row and form a multiset over the ordered pairs (m, c). In cases where we are able to detect rank columns, we keep them as singletons and do not form such pairs. By disallowing arbitrary cell permutations in the column matching, we avoid spurious matches, which are prone to occur with columns over widely used domains (e.g., dates and locations) and numerical values with unspecified units (ranks, ages, etc.). For example, two tables giving infant mortality percentages by country, one using CIA estimates and the other using World Bank estimates, would have a lower similarity from this because any coincidentally equal percentages would not find a match unless they happened to co-occur with the same country.

The similarity is computed as follows. Let i (for "intersection") denote the total number of pairs matched in this process. Then we take $sim(S, T) := \frac{i}{\max\{|S|, |T|\}}$ where $|S|$ and $|T|$ denote the number of cells in S and T, respectively. An alternative is to take $sim(S, T) := \frac{i}{|S|+|T|-i}$, so as to divide the intersection size by the union size; both gave similar results in our experiments.

Example: Figure 3 illustrates a source table S on the left and a target table T on the right. The member columns, $S1$ of S and $T2$ of T, are indicated using boldface. Value pairs formed from cell values in $S2$ anchored with associated values from the member column are $\{(C, 1), (A, x), (B, y)\}$, and for $S3$ are $\{(C, 2), (A, y), (B, z)\}$. Similarly, for columns $T1$ and $T3$ of T they are $\{(B, 2), (C, 3)\}$ and $\{(B, y), (C, z)\}$, respectively. A maximum weight matching has edges $(S1, T2)$, for the matching of member columns, with weight 2; $(S2, T3)$ with weight 1; and $(S3, T1)$ with weight 0. We remove the edge $(S3, T1)$ since it is zero-weight. The total sum of weights is 3 and, therefore, the similarity score is $3/9$ since the larger table, S, has $|S| = 9$ total cell values. □

S

S1	S2	S3
C	1	2
A	x	y
B	y	z

T

T1	**T2**	T3
2	**B**	y
3	**C**	z

Figure 3: Table similarity example: source and target tables with $sim(S, T) = 3/9$

For a fixed table at the previous epoch in time, we measure the similarity to all tables from the same page at the current epoch to determine which one, if any, is its counterpart.[15] For cases where multiple tables have the same (highest) similarity score, we break ties by considering table schema. However, if none of the tying tables has the same schema, or if more than one of them does, ties are then further broken by considering the section name hierarchy from the page's table of contents, for pages containing one, and choosing the table having longest subpath match with regards to the section hierarchy. This is especially useful on pages where the same set of entities is ranked according to different criteria in different tables such as Mountain Peaks in Hawaii[16] which includes tables ranked by `Elevation`, `Prominence` and `Isolation`, all having a similarity of 1 with one other.

Naturally, there is still some potential for the similarity score to choose the wrong table, and we shall evaluate the performance in Section 6. We have some heuristics that can be used to detect when this happens, including the approach based on text surrounding the table on both sides modulo section breaks. While such text is too brittle for table tracking, it can be useful for alarming suspicious behavior when the surrounding text of paired tables disagrees while some other table from the same page uniquely has the same surrounding text. Finally, we note that a more recent version of a table can be inserted without the now stable table being removed. For example, assume we are tracking Richest Companies (2017 Statistics) and Richest Companies (2018 Statistics) is added to the page. Ideally, the algorithm should ignore the former and start using the latter. Our current algorithm does not handle such (rare) cases and we employ additional techniques that are beyond the scope of this paper.

5.2 Alarm Policy

We do not expect that dynamic maintenance of templates can be fully automated and instead provide a solution that tries to minimize the burden of manual inspection and revision. Table tracking is thus applied using a three-tiered

[15] We test for rare cases where the page URL changes between epochs by looking up the canonical URL for each page.

[16] https://en.wikipedia.org/wiki/List_of_mountain_peaks_of_Hawaii

Table 4: Interestingness ("Very", "Somewhat", "Not") percentages, and percentage with no majority ("NoMaj"), for various fun fact generation methods

Method	Very	Somewhat	Not	NoMaj	W/L
simple top-k	53.7	37.4	0	8.9	(*)
selective top-k	48	41	3	8	
distributional	27	64	13	14	
top-k combo	54	35	1	10	
distrib combo	45	38	2	14	
trivia-quiz [19]	13.5	60.4	15.6	10.4	N/A
er-desc [24]	9.4	43.5	38.8	8.2	3.75:1
factoid-subj [18]	1.1	25.3	68.4	5.3	79:1
factoid-obj [18]	4.2	34.7	56.8	8.8	41:1
categories [23]	4	39	50	7	42:1

policy based on the similarity score of a table across two consecutive epochs: those above some *safe* threshold are successfully mapped; those below some *unsafe* threshold are not mapped (and, hence, are temporarily suppressed from triggering facticles) along with a (non-urgent) alert; and those falling in between, which come with an message indicating that further examination is needed. These thresholds should be set based on application-specific risk levels of displaying incorrect content vs coverage loss, but also based on analysis of the trade-offs with respect to false-positives and false-negatives. This question is investigated further in Section 6; in practice, we used 0.1 and 0.5 for the unsafe and safe thresholds, respectively.

5.3 Spam Protection

While we are restricted from providing exact details, we mention a few concepts that are useful for preventing malicious users from vandalizing Wikipedia content to affect facticles. First, there are blacklists in place (per-row and per-table) to quickly take down questionable content. Second, any new table content, such as new entities or attribute values associated with an existing entity, is sufficiently delayed from being used for a facticle, to give Wikipedia editors and spam bots a chance to detect and correct improper content. Facticles have now been in production for almost a year and there has been no issue with spam.

6 EXPERIMENTS

Below we demonstrate that facticles are liked by users, show the efficacy of automatically generating templates, and investigate the ability to successfully maintain pairing of templates with their corresponding tables over time.

6.1 Interestingness of Fun Facts

Using an (internal) production crowd evaluation platform, workers were presented with a query entity name, some background about the entity, and a statement about the entity. For example, the entity *Frozen* was presented with the background sentence, "*Frozen* is a 2013 American 3D computer-animated musical fantasy film produced by Walt Disney Animation Studios" and the facticle, "*Frozen* is the highest-grossing animated film of all time." Workers were asked, "How interesting is this statement to a user who is querying for the entity?" and given a rating scale ("Not Interesting", "Somewhat Interesting", "Very Interesting"). For each item, workers were selected from a large rater pool (many thousands) such that each rater was restricted to evaluating at most 6 items in total; hence, different items were rated by a different sets of workers. Table 4 summarizes the results for facticles based on different views in this section, as well as for various baselines which are discussed in Section 6.2. The numbers reported in the table are the percentage of items for each majority rating level ("Very", "Somewhat", "Not") as well as the percentage of items for which there was no majority among raters ("NoMaj"). The "W/L" column only pertains to Section 6.2 and is explained there.

For each of the 1000 entities that were sampled based on query stream frequency and on having an associated facticle, 5 workers rated interestingness. Very few of the facticles were considered not interesting (none for simple top-k, 3% for selective top-k and 13% for the distributional view). We believe the lower interestingness ratings observed for the distributional view was due to the facticle pivoting towards the category entity, and thus not being directly relevant to the query entity, rather than inherent interestingness. For example, the facticle "2 of the 5 tallest buildings in the world are in China, including Shanghai Tower" is arguably more about China than Shanghai Tower. We tried 100 examples that were rephrased to put more emphasis on the query entity, e.g., "Shanghai Tower is among the 2 of the 5 tallest buildings in the world that are in China", but the results were similar.

We also examined the interestingness of template combinations stitched together using an "as well as" clause for the same entity, as described in Section 3.3, using 100 entities. The results for simple top-k view combinations were very similar to their single-clause counterparts but for distributional view combinations the interestingness increased; see Table 4. We further ran a side-by-side to understand the marginal value added by combinations (i.e., single-clause facticle vs combo facticle based on the same entity), and gave workers the option to choose from "Left is better", "Right is better" and "About the same"). Workers preferred the combination views by a ratio of 4:1 for the simple top-k view and 2:1 for the distributional view.

Finally, we evaluated the interestingness of facticles in comparison to attribute-value facts shown in Google's Knowledge Panels. For example, facts for *Frozen* include `Release date`

and `Directors`. A randomly chosen fact from among these was highlighted and workers were asked whether it would be better for a facticle instead of the highlighted fact to appear in the Knowledge Panel, presented as a side-by-side using 200 entities and 3 workers (out of the pool of hundreds) per entity. The ratio of cases in which a facticle was preferred was 2.72:1.

6.2 Comparison to Existing Methods

For direct comparisons, we presented a side-by-side, with sides randomly swapped, to 3 workers and asked them to select from "Left is better", "Right is better' and "About the same"; win-loss gives the ratio of items for which a majority preferred simple top-k facticles over the baseline. We were limited by overlap size in comparing with baselines using the same set of entities. Therefore, Table 4 reports direct comparisons (in the "W/L" column) when possible as well as (one-sided) interestingness levels to allow for indirect comparisons. Facticles, for all its view types, had a higher interestingness level than all five baselines discussed below.

We directly compared against the category-based fun facts (`categories`) from [23]. Using a list of 400 most popular Wikipedia entities borrowed from [23], we selected 200 entities for which both the category-based approach and ours yielded a statement. For 84% of the entities, a majority of raters preferred the facticles, whereas for 2%, a majority preferred the category-based fun facts, a win-loss ratio of 42:1. (There was no preference for 8% and no majority for the remaining 6%.) We also compared against entity-relationship descriptions (`er-desc`) generated via templates for Freebase triples in [24] using 100 common entities. For 45% of the entities, a majority preferred the facticles, whereas for 12% a majority preferred entity-relationship descriptions, a win-loss ratio of 3.75:1. (There was no preference for 33% and no majority for the remaining 10%.)

In addition, we compared facticles against approaches that generate natural language questions from knowledge graph triples using a neural language model. The approach in [18] chooses triples randomly, such as (*Texas_Speedway*, *contained_by*, *Denton_County*) to generate the question, "Where is Texas Motor Speedway?"; we employed techniques from [6] to transform these questions into answers based on substituting the wh- phrase ("who", "which musician", etc.) with the object entity. Because the directionality of predicates in triples varies (e.g., some triples use *contained_by* whereas others use *contains*), we performed two separate evaluations using the subject and object for the query entity, presented in Table 4 as `factoid-subj` and `factoid-obj`, respectively. We also performed a side-by-side against facticles for a random subset of triples where both the subject and object have an associated facticle. We compared the same factoid statement using subject and object as the query entity with two respective facticle statements and took the best of the two ratings for the factoid. Even with this advantage, workers preferred facticles for 75% of the cases. Finally, we compared against the approach to generate quiz questions in [19], where the emphasis is on selecting triples that yield more difficult, hence more interesting, questions. Unfortunately, the code is proprietary and the released data only contains 3,411 questions based on a total of 76 entities, only 5 of which overlap with facticles. So we were unable to perform a side-by-side evaluation. However, we present interestingness ratings using a random sample of 100 questions transformed into statements in Table 4 as `trivia-quiz`.

Table 5: Superlative phrase prediction accuracy

Method	Test accuracy
Majority class ("tallest")	13.9%
Using Title Only	43.2%
Proposed Model	80.5%
Proposed Model w/ Substitution	86.3%

6.3 Template Generation and Refinement

We used a random forest classifier with hyper-parameters (number of trees, etc.) set based on 10-fold random train/test split cross-validation.

Superlative phrase prediction. Table 5 summarizes the test set accuracy based on 1117 raw top-k templates that were edited by native English speakers and annotated with the superlative phrase component. Overall, we found that our language model predicted the exact superlative phrase with 80.5% accuracy. In some cases where the prediction did not exactly match the training example, the predicted superlative phrase was interchangeable. For example, "largest by population" was in the template but the model predicted "most populous", which can reasonably be substituted. Similar cases include "tallest" vs. "highest" and "wealthiest" vs. "richest". By accounting for these, the accuracy further improves to 86.3%. The remaining errors were mostly due to unseen, typically long-tail, descriptions like "nearest terrestrial exoplanet candidates" and "most recent supercontinents".

Phrasal verb prediction. Table 6 summarizes the test set accuracy based on 167 raw selective templates that were edited by native English speakers and annotated with the phrasal verb component. We can see that the inference model gave roughly 71% accuracy. Again in some cases the mis-predicted phrases were interchangeable (e.g., "are in the state of" vs. "are in", "are in the canton of" vs. "are in"); and with these the accuracy improves to 73.8%. Similar to superlative prediction, the errors were mostly due to unseen, long-tail patterns such

Table 6: Phrasal verb prediction accuracy

Method	Test accuracy
Majority Class ("are in")	40.7%
Proposed Model	71.3%
Proposed Model w/ Substitution	73.8%

as `[Program]` "are broadcast during" `[Broadcast_time]`, and `[Manga]` "are marketed to" `[Demographic]`.

Both of these two components are needed for distributional view templates. The *combined* prediction accuracy of these components was 60.9%, which means that there's a strong correlation between individual component accuracies. This is beneficial from the standpoint of post-hoc manual editing time because the marginal work to correct multiple components compared to a single one is relatively small. To ascertain correctness, unfortunately all the templates must be inspected since the model cannot predict with certainty; but sorting the templates by prediction probability score is a useful way to triage the workload.

To evaluate the user satisfaction resulting from generated natural language, we used the predicted superlative and phrasal verb components to generate distributional facticles and compared this against the same facticles using a baseline template with `have [$category_name] [$category_value]` in place of the phrasal verb clause. Raters were presented with these alternatives in a side-by-side and asked to select which side is better, or "About the same". There were 113 such alternatives compared and each was evaluated by three raters. The phrasal verb facticle was preferred over the baseline 33.6% of the time and the baseline was preferred only 0.9% of the time. The remaining cases were "About the same" (61.9%) and "No majority" (3.5%).

6.4 Dynamic Maintenance

We tracked approximately 1500 superlative tables across 74 consecutive epochs with a gap in time such that there were two epochs per week, hence, 37 total weeks. Ground truth data for the semantic counterparts of tables across epochs does not exist. Since examining all the tables at all consecutive epochs to judge this is very time-intensive, we employed the following heuristic. For each table, we consider only the first and last epochs of the tracking period, based on the assumption that if a table goes "off the rails" at some epoch due to a spurious match with the wrong table, then it is unlikely to "come back" at a later epoch. That is, if a table at the first epoch tracks to some table at the last epoch that is indeed its counterpart, then we should expect tracking to have been successful at the intermediate epochs. The process for examining two tables to determine if they are semantic counterparts is as follows. First, verbal descriptions are formed from the respective tables based on the entity class and superlative, plus any spatio-temporal scope (e.g., "tallest buildings in New York City") or domain-specific filtering predicates (e.g., "tallest buildings that are considered towers") indicated by the containing section name, table caption or surrounding text on the page. These descriptions should be synonymous but precise.[17]

No other table from the respective pages should have a closer matching such description than the two candidate tables that are being considered as counterparts.

92% of the tables tracked "survived" to the last epoch, with similarity scores above the "unsafe" threshold of 0.1 across all epochs. There are 8 false-positives in those survived (3 of which have scores above the *safe* threshold of 0.5) and 17 false-negatives in those did not.

For similarity scores between the safe and unsafe thresholds, our system files alerts. Luckily, these alerts were infrequent, occurring in only 0.15% of the table-epochs for an average of 3 total alerts per week. The 3 false-positives above the safe threshold were due to a particularly tricky case where a long table was split into multiple ones. Figure 4 gives an example of one of these, which ranks the longest-running Indian TV Series by number of episodes. The table was broken out into one small table listing the top few row (with over 2,000 episodes) and one large table listing the remaining ones. The similarity was higher for the larger table due to more overlap. Fortunately such cases are rare.

A representative example of the 17 false-negatives (i.e., the source table from the first epoch had a counterpart table from the same page in the last epoch that the similarity score was too low to be tracked) is shown in Figure 5. Both tables rank the largest protected areas in the world by size in square kilometers and are, in fact, the only table on their respective pages. There was a single intermediate epoch at which the table was truncated from having 157 rows to maintaining only the top-20 and many of the member column cell values were significantly changed. As a result, these tables were temporarily dropped from production and an alert was issued. Note that the schema of these two tables also changed quite a bit, with a new column added and several columns renamed and/or broadened, so a schema matching based approach would have lost track due to the changes.

To compare against using the table sequence number to track tables, we examined the sequence numbers of the true-positives at the begin and end epochs and observed that approximately 14.5% (199) changed, meaning that they would have been false-negatives. We also compared against using table schema to track tables and found that there were 18

[17] E.g., we distinguish between "tallest" and "highest" mountains.
[18] Links to epochs: before; after.
[19] Links to epochs: before; after.

S: Longest-running Indian TV series (Over 1,000 episodes); 59 rows			
Title	Network	Language	Episode Count
Yeh Rishta Kya Kehlata Hai	Star Plus	Hindi	2,429
Balika Vadhu	Colors TV	Hindi	2,248
...	...	...	...
Cinemala	Asianet	Malaylam	1,000+
Ek Mahal Ho Sapno Ka	Sony	Hindi	1,000+

T_1: Longest-running ... (Over 2,000 episodes); 4 rows			
Title	...	...	Episode Count
Yeh Rishta Kya Kehlata Hai	...	...	2,434
Taarak Mehta Ka Ooltah Chashmah	...	...	2,250
Balika Vadhu	...	...	2,248
Saath Nibhaana Saathiya	...	...	2,182

T_2: Longest-running ... (1,000-1,999 episodes); 55 rows			
Title	...	...	Episode Count
Sasural Simar Ka	...	...	1,925+
Kyunki Saas Bhi Kabhi Bahu Thi	...	...	1,833
...	...	...	...
Cinemala	...	...	1,000+
Ek Mahal Ho Sapno Ka	...	...	1,000+

Figure 4: Table tracking false-positive example: S got mapped to T_2 but actually was split into T_1 and T_2[18]

S: 157 rows			
Name	Country	Size (square kilometers)	Date established
Ross Sea Marine Reserve	Antarctica	1,550,000	December 2017 ...
PapahĀĄnaumokuĀĄkea Marine National Monument ...	United States ...	1,510,000,	2006
...	...	...	...
NouabalÃĽ-Ndoki National Park	Republic of Congo	4,000	1993
ROSCI0227 ...	Romania	3,600	2008

T: 20 rows				
Rank	**Name**	Country or area	Size (sq. km)	Year designated
1	**Marae Moana**	Cook Islands	1,976,000	2017
2	**Ross Sea Region Marine Protected Area**	Antarctica	1,555,851	2017
...	...	...	...	...
19	**Offshore Trap/Pot Waters Area**	Western Atlantic Ocean, United States	336,101	1997
20	**Nazca-Desventuradas Marine Park**	Desventuradas Islands, Chile	300,035	2016

Figure 5: Table tracking false-negative example: S went through a major clean-up and condensing between two epochs to become T; $sim(S, T)$ was only 0.03[19]

false-positives and 305 false-negatives. Both are significantly worse than our tracking algorithms.

7 CONCLUSIONS

In this paper, we presented an approach to generate natural language trivia statements, or fun facts, from superlative tables on Wikipedia pages. Our approach is templated and relies on automatic instantiation with (template, table) pairs; this is crucial for keeping the fun facts up to date. We designed ML models to automatically generate various critical template components, including the *superlative phrase* and *phrasal verb*, for the two view classes we propose. We introduced a table tracking method to dynamically maintain the (template, table) pairing across time. Our experiments showed that (1) fun facts generated using our proposed classes are interesting to users and are liked much more than existing generative approaches; (2) our models for template generation is accurate compared to manually curated content and the natural language that it generates increases user satisfaction; and (3) table tracking reduces the manual maintenance burden as it is resistant to false-positives and results in a small enough number of false-negatives that lightweight manual post-processing can correct for tables that are improperly dropped.

Future work could include the extension to other (non-Wikipedia) tables on the Web and CSV files, for which rich metadata that is currently relied upon may not exist; and the design of additional template classes (e.g., to capture outliers) that could also apply to tables other than superlative tables.

REFERENCES

[1] Nicholas D. Allen, John R. Templon, Patrick Summerhays McNally, Larry Birnbaum, and Kristian J. Hammond. 2010. StatsMonkey: A Data-Driven Sports Narrative Writer. In *Computational Models of Narrative, Papers from the 2010 AAAI Fall Symposium, Arlington, Virginia, USA, November 11-13, 2010*. http://www.aaai.org/ocs/index.php/FSS/FSS10/paper/view/2305

[2] Sreeram Balakrishnan, Alon Y. Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. 2015. Applying WebTables in Practice. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*.

[3] Michael J. Cafarella, Alon Y. Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. 2018. Ten Years of WebTables. *PVLDB* 11, 12 (2018), 2140–2149.

[4] Fernando Chirigati, Jialu Liu, Flip Korn, You Wu, Cong Yu, and Hao Zhang. 2016. Knowledge Exploration using Tables on the Web. *PVLDB* 10, 3 (2016), 193–204. http://www.vldb.org/pvldb/vol10/p193-chirigati.pdf

[5] Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to Ask: Neural Question Generation for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. 1342–1352. https://doi.org/10.18653/v1/P17-1123

[6] Michael Heilman and Noah A. Smith. 2010. Good Question! Statistical Ranking for Question Generation. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*. 609–617. http://www.aclweb.org/anthology/N10-1086

[7] Roy Jonker and A. Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* 38, 4 (1987), 325–340. https://doi.org/10.1007/BF02278710

[8] Mitesh M. Khapra, Dinesh Raghu, Sachindra Joshi, and Sathish Reddy. 2017. Generating Natural Language Question-Answer Pairs from a Knowledge Graph Using a RNN Based Question Generation Model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. 376–385. http://aclanthology.info/papers/E17-1036

[9] Andreas Kokkalis, Panagiotis Vagenas, Alexandros Zervakis, Alkis Simitsis, Georgia Koutrika, and Yannis E. Ioannidis. 2012. Logos: a system for translating queries into narratives. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*. 673–676. https://doi.org/10.1145/2213836.2213929

[10] Georgia Koutrika, Alkis Simitsis, and Yannis E. Ioannidis. 2010. Explaining structured queries in natural language. In *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*. 333–344. https://doi.org/10.1109/ICDE.2010.5447824

[11] Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. 1203–1213. http://aclweb.org/anthology/D/D16/D16-1128.pdf

[12] Matthew Merzbacher. 2002. Automatic Generation of Trivia Questions. In *Foundations of Intelligent Systems, 13th International Symposium, ISMIS 2002, Lyon, France, June 27-29, 2002, Proceedings*. 123–130. https://doi.org/10.1007/3-540-48050-1_15

[13] James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics* 5, 1 (1957), 32–38.

[14] Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. 1470–1480. http://aclweb.org/anthology/P/P15/P15-1142.pdf

[15] Rivindu Perera and Parma Nand. 2017. Recent Advances in Natural Language Generation: A Survey and Classification of the Empirical Literature. *Computing and Informatics* 36, 1 (2017), 1–32. http://www.cai.sk/ojs/index.php/cai/article/view/2017_{11}

[16] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering Table Queries on the Web using Column Keywords. *PVLDB* 5, 10 (2012), 908–919. http://vldb.org/pvldb/vol5/p908$_{\text{r}}$akeshpimplikar$_{\text{v}}$ldb2012.pdf

[17] Abhay Prakash, Manoj Kumar Chinnakotla, Dhaval Patel, and Puneet Garg. 2015. Did You Know? - Mining Interesting Trivia for Entities from Wikipedia. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. 3164–3170. http://ijcai.org/Abstract/15/446

[18] Iulian Vlad Serban, Alberto García-Durán, Çaglar Gülçehre, Sungjin Ahn, Sarath Chandar, Aaron C. Courville, and Yoshua Bengio. 2016. Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1056.pdf

[19] Dominic Seyler, Mohamed Yahya, and Klaus Berberich. 2017. Knowledge Questions from Knowledge Graphs. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR 2017, Amsterdam, The Netherlands, October 1-4, 2017*. 11–18. https://doi.org/10.1145/3121050.3121073

[20] Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. 2015. Incremental Knowledge Base Construction Using DeepDive. *PVLDB* 8, 11 (2015), 1310–1321. https://doi.org/10.14778/2809974.2809991

[21] Alkis Simitsis, Georgia Koutrika, Yannis Alexandrakis, and Yannis E. Ioannidis. 2008. Synthesizing structured text from logical database subsets. In *EDBT 2008, 11th International Conference on Extending Database Technology, Nantes, France, March 25-29, 2008, Proceedings*. 428–439. https://doi.org/10.1145/1353343.1353396

[22] Professor Steven Skiena and Charles Ward. 2013. *Who's Bigger?: Where Historical Figures Really Rank*. Cambridge University Press, New York, NY, USA.

[23] David Tsurel, Dan Pelleg, Ido Guy, and Dafna Shahaf. 2017. Fun Facts: Automatic Trivia Fact Extraction from Wikipedia. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*. 345–354. http://dl.acm.org/citation.cfm?id=3018709

[24] Nikos Voskarides, Edgar Meij, and Maarten de Rijke. 2017. Generating Descriptions of Entity Relationships. In *Advances in Information Retrieval - 39th European Conference on IR Research, ECIR 2017, Aberdeen, UK, April 8-13, 2017, Proceedings*. 317–330. https://doi.org/10.1007/978-3-319-56608-5_25

[25] Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in Data-to-Document Generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. 2253–2263. https://aclanthology.info/papers/D17-1239/d17-1239