Knowledge Exploration Using Tables on the Web

Fernando Chirigati[†*] Jialu Liu[§*] Flip Korn[§] You (Will) Wu[§] Cong Yu[§] Hao Zhang[§]
[†]New York University [§]Google Research NYC
fchirigati@nyu.edu {jialu,flip,wuyou,congyu,haozhang}@google.com

ABSTRACT

The increasing popularity of mobile device usage has ushered in many features in modern search engines that help users with various information needs. One of those needs is Knowledge Exploration, where related documents are returned in response to a user query, either directly through right-hand side knowledge panels or indirectly through navigable sections underneath individual search results. Existing knowledge exploration features have relied on a combination of Knowledge Bases and query logs.

In this paper, we propose Knowledge Carousels of two modalities, namely sideways and downwards, that facilitate exploration of IS-A and HAS-A relationships, respectively, with regard to an entity-seeking query, based on leveraging the large corpus of tables on the Web. This brings many technical challenges, including associating correct carousels with the search entity, selecting the best carousel from the candidates, and finding titles that best describe the carousel. We describe how we address these challenges and also experimentally demonstrate through user studies that our approach produces better result sets than baseline approaches.

1. INTRODUCTION

Driven in part by increased mobile phone usage, modern search engines have been striving to enhance the search experience beyond ten blue links. The goal is to anticipate user needs to help get information as effortlessly as possible. Examples of recent features [19, 21] in various search engines (*e.g.*, Baidu, Bing, Google, and Yahoo!) are: *Knowledge Cards*, which summarize various facts about the query entity; "*People also search for*," which lists entities that often co-occur with the query entity; and curated related entities for specific verticals, *e.g.*, "Movies and TV Shows" for actors.[1]

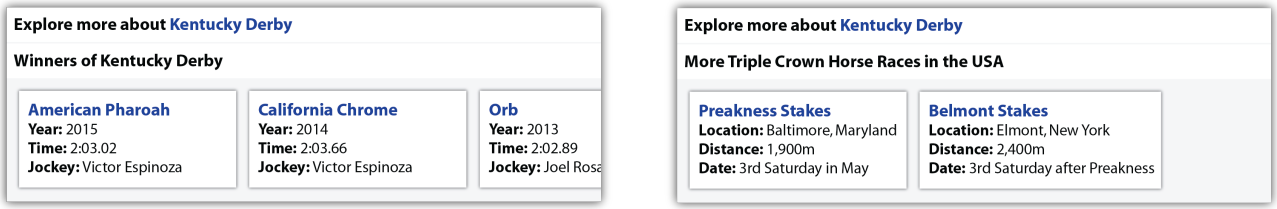
These features, often powered by a Knowledge Base and/or query logs [4, 16, 22, 26], increasingly aim to help users who are interested not just in a specific answer but in a general topic, given that between 30-50% of all queries are entity-seeking [13, 18] and, therefore, allow exploration of the search entity's attributes and its relationships to other entities. However, current features lack context for the related entities presented. For example, "*People also search for*" for the query [barack obama] in Google mixes together conceptually different groups of people including Vladimir Putin (head of state), Michelle Obama (family member), and Hillary Clinton (cabinet member / former Senator) without any context, let alone consistency, for *how* they are related. Other features are highly curated and/or tailored to specific verticals such as providing the cast for a movie entity.

Our approach to this problem aims to facilitate exploration along two query modalities, called *downwards* and *sideways*, that correspond to HAS-A and IS-A relationships, respectively. Downwards provide highlights of an entity with respect to some facet (*e.g.*, the oeuvre of an artist, or the participants of an event), while sideways provide potentially useful associations (*e.g.*, technology companies like Google, or Republican presidential candidates besides Donald Trump), enabling comparisons between peers. Figure 1(a) shows an example of a downward for the query [kentucky derby] and Figure 1(b) shows a sideway for the same entity. We call this presentation a *Knowledge Carousel* [2] and each of these is potentially interesting and useful for exploring the original query. The downward presents winning horses from various years of the Kentucky Derby, along with facts from shared attributes such as year of event and finish time; the sideway presents horse races associated with the Kentucky Derby, namely, Belmont Stakes and Preakness Stakes (which together form the Triple Crown of Thoroughbred Racing), along with facts such as location and time of year.

Downwards are, to some extent, analogous to *properties* from Knowledge Bases.[3] For example, the Kentucky Derby entity not only includes properties such as *name* and *website* but also those that connect it to other entities, *e.g.*, specific instances of the race from individual years via */time/recurring_event/instances*, each of which is connected to a winning horse entity via */award/competition/winner*. However, while Knowledge Bases include a vast number of entities, they lack idiosyncratic or *long-tail* properties, and those that do exist are inconsistent across different entities of the same type [10]. Sideways are analogous to sets of entities having the same *type* (and the ontology of *domains*

[*]Work done during internship.
[1]These three features are collectively known as *Knowledge Panels* in Google and *Satori* in Bing.
[2]The term *carousel* refers to the visual layout of members that can be swiped to the left or right on a mobile phone.
[3]Examples from Freebase.

Proceedings of the VLDB Endowment, Vol. 10, No. 3



(a) (b)

Figure 1: Example of knowledge exploration for the query [kentucky derby] through Knowledge Carousels: (a) a downward showing the winners of Kentucky Derby; (b) a sideway representing the famous Triple Crown horse races in the US, of which Kentucky Derby is a member.

containing these types) from Knowledge Bases, though types tend to be very broad. For example, the Kentucky Derby entity belongs to *recurring_event*, *recurring_competition* and *literature_subject* but does not include specialized categories such as Triple Crown Horse Races.[4]

Obstacles to being able to assemble related entities with associated context using only a Knowledge Base are well known: it is for this reason that Knowledge Bases are used for multi-entity query answering only when a suitable table cannot be found [25]. Therefore, we make use of the large corpus of tables on the Web (WebTables) and argue that, for the aforementioned query types, this approach works well for the following reasons. First, tables are highly structured and, as a result, related entities are easy to find since they exist within the same column. Second, tables are often highly curated with explicit contextual information (*e.g.*, header attributes, title, and surrounding text) that is very useful in understanding the concepts associated with the entities. Third, the table structure allows for inferring implicit features (*e.g.*, semantic types and subject columns) by reasoning across columns. Finally, they can leverage query logs associated with the Web page on which the table appeared.

In short, Knowledge Bases tend to be geared towards understanding single entities whereas WebTables contain groups of related entities and, therefore, require less assembly to produce downwards or sideways from them. Nevertheless, transforming WebTables into carousels is highly non-trivial, *e.g.*, only some tables make for a good carousel, specifically, those that can be compactly represented by members and facts, and it can be tricky to find the member column and extract a single entity from its cells as members. In this paper, we describe manifold challenges of Knowledge Exploration via tables on the Web, including: (i) selecting WebTables that make for a good carousel presentation and carefully choosing members and facts for this presentation; (ii) finding carousels relevant to a given entity; (iii) ranking these relevant carousels; and (iv) generating a concise and human-understandable title. As we shall see in Section 4.4, a good title is important for context by describing how member entities of the carousel are related to the search entity. We discuss our current solutions for each of these issues and evaluate them through different user studies that confirm the usefulness of WebTables for Knowledge Exploration.

Overall, our main contributions are as follows: (i) a novel approach of using WebTables for generating Knowledge Carousels, its formal problem definition, and the categorization of two important exploration modalities, *downwards* and *sideways* (Section 3); existing search features provide carousels that bear conceptual similarity, but they are geared to specific verticals and highly curated (*e.g.*, "Cast" for a given movie could be considered a downward; "Action Movies" could be considered a sideway), whereas the methods we propose are fully-automated and horizontal; (ii) solutions for the main technical challenges: matching carousels to the search entity, ranking candidate carousels, and generating titles (Sections 4 and 5); and (iii) extensive user studies evaluating our solutions against alternative approaches and demonstrating that we can provide high quality carousel exploration, outperforming state-of-the-art baselines (Section 6).

2. RELATED WORK

Recently, various search engines have provided users with features that have the flavor of entity-based knowledge exploration; a prominent example of this trend is "*People also search for*" based on combining query logs, Knowledge Bases and, to a lesser extent, social media.[5] However, lack of consistency and semantic drift are known problems with the entities recommended [26]. In [4, 14], member entities are constrained to be of the same type and ranked via a regression model based on click co-occurrence and PageRank. Since Yahoo! Spark is no longer in production, we are unable to test the available types but, based on those in their examples (*e.g.*, "Related People" and "Related Movies" for query [jennifer aniston]), these techniques would still suffer from semantic drift. More recent search features in Google's Knowledge Panel and Bing's Satori provide sideway and downward carousels for entities in a limited set of verticals, such as movies, and the specific concepts within those concepts, such as "Cast" or "Other Action Movies", are manually curated. The approach we propose in this paper is automated and, therefore, horizontal.

One of the important aspects of a knowledge carousel is finding member entities. A widely used approach to find related entities is *concept set expansion*: given an initial set of seed entities, scour the Web for additional entities occurring in similar contexts (*e.g.*, via Hearst patterns) and, thus, potentially belonging to the same concept. Leveraging column structure from WebTables for concept set expansion has in fact been shown to reduce semantic drift [8, 23]. All these methods require several seed entities for meaningful set expansion.

Related but orthogonal problems that have been studied include table search [2, 17], table-based query answering [2,

[4]More popular entities have a richer set of types; for example, *Barack Obama* includes more than 80 types from over 25 domains.

[5]Our experiments in Section 6.2 try to capture this feature as the Knowledge Base method.

Kentucky Derby winners[19]

Year	Winner	Jockey	Trainer	Owner	Time
2015	American Pharoah †	Victor Espinoza	Bob Baffert	Zayat Stables, LLC	2:03.02
2014	California Chrome	Victor Espinoza	Art Sherman	Steve Coburn & Perry Martin	2:03.66
2013	Orb	Joel Rosario	Claude McGaughey III	Stuart S. Janney III & Phipps Stable	2:02.89
2012	I'll Have Another	Mario Gutierrez	Doug O'Neill	J. Paul Reddam	2:01.83

Figure 2: A WebTable describing the winners of the Kentucky Derby horse race.

25], subgraph similarity search in Knowledge Bases [16], and query aspect identification [24]. Of these, table search is perhaps the most similar in that the user submits a query and gets back a ranked set of relevant tables. One crucial difference is in terms of user intent. Table search is a retrieval problem where the user has a specific information need that can be answered by matching tokens (*e.g.*, in the table header, subject column, or table context as in [2, 5]) whereas carousels are for (sideways and downwards) exploration. Therefore, features (*e.g.*, ignores popularity) and methodology (*e.g.*, training labels obtained pointwise) for ranking in table search are not applicable to carousel ranking. Section 6.3 demonstrates this with an experimental comparison using a state-of-the-art table search system to rank carousels.

3. OVERVIEW

In this section, we provide an overview of our WebTable driven approach to facilitate Knowledge Exploration. Our exposition focuses specifically on the so-called *horizontal* WebTables [2] from the English Wikipedia. The reason for this is threefold: these tables are often of higher quality compared to other tables on the Web due to Wikipedia collaborative editing policies; they are rarely duplicated across multiple pages; and the English language corpus is the largest of all languages. We leave it for future work to generalize these techniques to other tables on the Web.

3.1 WebTables

We begin by introducing a formal definition of WebTables [5], *i.e.*, an HTML table extracted from a Web page.

Definition 1 (WebTable). A WebTable extracted from a Web page p is a tuple $\mathbb{w} = (r_h, R, A, Q)$, where

- $r_h = (a_1, a_2, ..., a_m)$, a.k.a. header row, is a sequence of attributes from a special row that is annotated as the header row of the table;
- $R = \{r_1, r_2, ..., r_n\}$ is the set of data rows from the table; for each $r_i \in R, r_i = (v_{i1}, v_{i2}, ..., v_{im})$ is a sequence of values with 1-to-1 correspondences to r_h;
- $A = \{c_1, c_2, ...\}$ is a set of auxiliary metadata extracted from p's HTML content about the table, where each element is a textual string; we specifically consider page title, table caption, contextual snippets before and after the table, and title of the content section containing the table;
- $Q = \langle (q_1, \rho_1), (q_2, \rho_2), ... \rangle$ is an ordered set of queries q_i that are generated from the click graph [6] using the Web search logs; the set is attached to p and ordered by their number of clicks ρ_i.

Figure 2 shows an example of a WebTable[6] for the entity *Kentucky Derby*. Here, r_h is the first row with attributes such as *Year*, *Winner*, and *Jockey*, and the remaining rows are data rows forming R. The caption "Kentucky Derby winners" on top of the table is part of the auxiliary metadata A. We use $\mathbb{W}$ to denote a set of WebTables.

In particular, the metadata associated with the table are extracted in the following ways: (i) page title is taken directly from the $\langle title \rangle$ tag of the page containing the table; (ii) table caption is taken directly from the $\langle caption \rangle$ tag associated with the table; (iii) contextual snippets are extracted from the HTML content surrounding the table, where we take up to 200 words (with HTML tags stripped) from before and after the table; and (iv) section title is extracted from the header (*e.g.*, $\langle h2 \rangle$) tag, if present, that is the closest before the beginning $\langle table \rangle$ tag.

Leveraging various previous works on extracting and annotating high quality WebTables [2, 5, 12, 17], we make the following assumptions about the table corpus that we employ in this paper. First, we assume that low quality tables (*e.g.*, tiny tables, calendar tables, and table-of-content tables [2]) are pruned away from the corpus and each table in the corpus is a good data table. Second, we assume that the header row of each table is properly detected and low quality non-data rows are discarded. While we are making those assumptions, it is important to note that our approach does not require 100% accuracy. In other words, our solution requires neither all of the tables to be of high quality (as long as most of the tables are of high quality), nor all header rows to be properly identified. Indeed, the table corpus that we use for our experiments has many mistakes in both aspects, and the experimental results are not affected.

3.2 Knowledge Carousels

Our goal is to generate the set of *Knowledge Carousels* from a large corpus of high quality WebTables to enable and/or assist Knowledge Exploration in Web search. Before we delve into the carousel generation problem, we formally define Knowledge Carousel.

Definition 2 (Knowledge Carousel). Given a set $\mathbb{E}$ of entities from a Knowledge Base, a Knowledge Carousel is a tuple $\mathbb{c} = (e, t, F, M)$, where

- $e \in \mathbb{E}$ is the *pivot entity* to which the carousel is associated;
- t is a human-readable title that describes the carousel and its association with the pivot entity;
- $F = \langle f_1, f_2, ..., f_k \rangle$ is the totally ordered set of fact names;
- $M = \langle m_1, m_2, ..., m_l \rangle$ is the totally ordered set of *carousel members*; each $m_i \in M$ is an entity and is associated with

[6]From `http://en.wikipedia.org/wiki/Kentucky_Derby`

a sequence of fact values $U_i = \langle u_{i1}, u_{i2}, ..., u_{ik} \rangle$; U_i has 1-to-1 correspondence to F; for simplicity of presentation and without loss of generality, we flatten any set-valued individual fact value into a single value.

Figure 1(a) illustrates an example of a Knowledge Carousel that can be derived from the WebTable example in Figure 2. Intuitively, the carousel allows the users to explore recent winners of the Kentucky Derby horse race. The type of the carousel is downward because *Winner* is semantically a property of *Kentucky Derby*, which is the pivot entity of the carousel. The members are *American Pharoah, California Chrome, Orb,* and so on, and the set of included facts are *Year, Time,* and *Jockey*. Similarly, Figure 1(b) shows an example of a sideway Knowledge Carousel for the same entity, with members being *Preakness Stakes* and *Belmont Stakes*, and the set of facts being composed of *Location, Distance,* and *Date*. For simplicity, we identify the entities based on their human-readable names in the examples. In the actual implementation, the entities are represented using machine IDs[7] to avoid naming conflicts, and translated back to names before being presented to users. Also, note that we only leverage the set of entities $\mathbb{E}$ from the Knowledge Base for named entity recognition, *i.e.*, we do not leverage the types or properties to which these entities might be associated inside the Knowledge Base.

We use $\mathbb{c}_e^d$ and $\mathbb{c}_e^s$ to denote a downward carousel and a sideway carousel, respectively, associated with pivot entity e. The superscripts and subscripts are dropped when they are obvious from the context or not needed. We use $\mathbb{C}$ to denote a possibly ordered set of Knowledge Carousels, and similar superscripts and subscripts apply to $\mathbb{C}$.

Intuitively, the quality of a Knowledge Carousel depends on the following main aspects. First, the members in the carousel should belong to a coherent semantic *concept* that is salient to the pivot entity. This saliency can be *downward*, *i.e.*, the members can collectively be considered a property of the pivot entity (*e.g.*, winners of a horse race or famous paintings housed at a museum). It can also be *sideway*, *i.e.*, the members are peers of the pivot entity and form a collection to which the pivot entity belongs and for which the pivot entity is known. For example, *Preakness Stakes* and *Belmont Stakes* are interesting peers to *Kentucky Derby* because those three races form the well-known collection of Triple Crown Races in the US. Second, a carousel must have a human-readable title that describes the concept of the carousel and its saliency with regard to the pivot entity. The task of generating the title has been overlooked in all previous exploratory search studies. However, this is of high importance, since, based on discussions with search product managers, the title is often the only visual cue that can grab the attention of the search users and motivates them to further explore.

3.3 Knowledge Carousel Generation

Each of the aforementioned quality requirements presents its technical challenges, particularly the requirement that the carousel should form a coherent concept that is salient to the pivot entity. Our core idea is that the corpus of WebTables allows us to achieve this better than previous approaches for the following reasons. First, many WebTables are painstakingly generated by real users who know and feel passionate about the subject.[8] As a result, they tend to naturally represent salient concepts that are interesting to people. Second, the tables are extracted from Web pages, which have a rich set of ranking signals that can be generated from the search logs. These signals can be leveraged to identify pages with the most interesting concepts. Third, there is a number of earlier works on annotating WebTables, which can be again leveraged for ranking the tables.

[7] http://wiki.freebase.com/wiki/Machine_ID

Table 1: List of notations for reference.

Notation	Explanation
Knowledge Carousels	
$\mathbb{C}, \mathbb{C}^d, \mathbb{C}^s$	set of carousels
$\mathbb{c}, \mathbb{c}^d, \mathbb{c}^s$	individual carousel
$\mathbb{c}_e, \mathbb{C}_e$	a (set of) carousel(s) for pivot entity e
$\mathbb{c}.t$	carousel title
$\mathbb{c}.F(U), \mathbb{c}.M$	fact names (values), and members
WebTables	
$\mathbb{W}$	the corpus of tables
$\mathbb{w}$	individual table
$\mathbb{w}.r_h$	header row, *i.e.*, list of attributes
$\mathbb{w}.R$	data rows in the table
$\mathbb{w}.v_{ij}$	value of j-th attribute of the i-th row
$\mathbb{w}.A, \mathbb{w}.Q$	table auxiliary metadata and queries

Having the WebTables corpus, however, is only the beginning. Constructing high quality Knowledge Carousels from the WebTables corpus still requires the solving of a suite of quality ranking challenges, which we will illustrate through the following formal problem definition.

Definition 3 (Knowledge Carousel Generation). Given a corpus of WebTables $\mathbb{W}$ and a set of entities in a Knowledge Base $\mathbb{E}$, for each $e \in \mathbb{E}$, generate a *possibly empty*, ordered set of Knowledge Carousels $\mathbb{C}_e$ with e as the pivot entity, such that:

- carousels associated with the same pivot entity are ranked according to a scoring function $\mathcal{S}$;

and for each carousel $\mathbb{c}_e \in \mathbb{C}_e$, $\exists \mathbb{w} \in \mathbb{W}$ such that:

- members are chosen from the *subject column* of r_h from among a subset of rows in R;
- facts associated with members are chosen from a subset of non-subject attributes in r_h.

Each Knowledge Carousel is a view derived from a WebTable by selecting a subset of attributes from the table header row as the facts and selecting a subset of the table data rows as the members and fact values. The pivot entity and the carousel members are mapped to a given Knowledge Base after named entities are recognized in the textual content of the table. Finally, the carousels are grouped by the pivot entities. Some entities may not be a pivot entity for any carousel, which will lead to an empty $\mathbb{C}$.

3.4 Challenges and Solution Overview

Despite the clear advantages of WebTables, there are various challenges in using them for Knowledge Exploration. First, members and facts must be correctly identified. Members are the subject of carousels, so the attribute that better reflects the subject of the WebTable must be detected. While

[8] Excluding those created purely for layout purpose, which are pruned away from the corpus.

one could assume that this attribute is the first one in the table, this is often not the case (see Figure 2, where *Winner* is the main attribute, and not *Year*). In addition, since a table can have a plethora of attributes, the most concise, interesting, and insightful ones must be chosen as facts to the carousel. We address these issues by having the notion of *critical* attributes: attributes are ranked based on their popularity and on how close they are from the subject of the table. Such novel ranking is then used to determine members and facts (Sections 4.1 and 4.2).

Second, in order to serve relevant carousel(s) for a search entity, we need to associate each carousel with the proper pivot entities, depending on the carousel type. Any member entity in a sideway carousel can serve as a pivot entity, because the member set collectively represents the desired concept. For downward carousels, the effort entails understanding which entities the table is about (for which entities the table potentially describes a property). We show how we use the context of a WebTable for this task in Section 4.3.

Third, we need a human-readable title that describes, in a few words, the subject being presented by a carousel. Even though we have a myriad of context elements in a WebTable that could be potentially used as titles, they all have drawbacks. For instance, most of the tables do not have captions, so leveraging these as titles is hardly feasible. While page titles have good coverage, they describe the entire page rather than a specific table, and multiple tables can be found in a single page. One could also use section titles, or a combination of these, but not all the tables can be identified as belonging to a section. Instead, in our approach, we leverage the query logs to generate concise and human-understandable titles (Section 4.4).

Last, for entities with more than one associated carousel, the core technical challenge is designing the scoring function $\mathcal{S}$ for the ranking. In Section 5, we show how we combine different elements, including *popularity* and *relatedness*, to design a ranking function that performs better than the state-of-art table ranking system (see Section 6.3).

An overview of the carousel pipeline is presented in Algorithm 1. We first generate pivot-less carousels ($\mathbb{C}$), one for each WebTable, and represent the many-to-many relationship between entities ($\mathbb{E}$) and these pivot-less carousels using a bipartite graph G, with two sets of edges: E^d (for downwards) and E^s (for sideways). Note that E^d and E^s are disjoint as we will discuss in Section 4.3. For each entity, we attach it as a pivot entity to all its neighboring pivot-less carousels in G, and rank its associated downward and sideway carousels, respectively.

We remark that choosing when to serve sideways or downwards and deciding the number of carousels to present are still open problems, and we leave these for future work.

4. CAROUSEL GENERATION

The carousel generation process relies on two important pieces of metadata associated with the WebTable—*critical* attributes and *subject* column—which are described in Section 4.1. The subsequent extraction of carousel members and facts is discussed in Section 4.2, and the identification of pivot entities from WebTable in Section 4.3. Following this procedure (line 1-13 in Algorithm 1), we are able to generate both downward and sideway carousels ($\mathbb{c}_e^d$ and $\mathbb{c}_e^s$, respectively) by grouping results based on each possible pivot entity e, each with a carousel title (Section 4.4).

Algorithm 1: Overview of the carousel pipeline.

Input: WebTables $\mathbb{W}$, Entities $\mathbb{E}$
Output: Carousel sets $\mathbb{C}^d, \mathbb{C}^s$

1 $\mathbb{C}^d \leftarrow \emptyset$; $\mathbb{C}^s \leftarrow \emptyset$
2 Bipartite graph $G = (\mathbb{E}, \mathbb{C}, E^d \cup E^s)$
3 $\mathbb{C} \leftarrow \emptyset$; $E^d \leftarrow \emptyset$; $E^s \leftarrow \emptyset$
4 **for** WebTable $\mathbb{w} \in \mathbb{W}$ **do**
5 create empty carousel $\mathbb{c}$
6 identify critical attributes and subject column (Section 4.1)
7 generate carousel members M from subject column (Section 4.2)
8 generate carousel fact names F and fact values U from columns of critical attributes (Section 4.2)
9 generate carousel title t (Section 4.4)
10 $\mathbb{C} \leftarrow \mathbb{C} \cup \{\mathbb{c}\}$
11 identify pivot entities for $\mathbb{c}$ as a downward carousel (Section 4.3.1) and as a sideway carousel (Section 4.3.2)
12 $E^d \leftarrow E^d \cup \{(e, \mathbb{c}) \mid e \in$ downward pivot entities$\}$
13 $E^s \leftarrow E^s \cup \{(e, \mathbb{c}) \mid e \in$ sideway pivot entities$\}$
14 **for** entity $e \in \mathbb{E}$ **do**
15 $\mathbb{C}_e^d \leftarrow \{\mathbb{c} \mid (e, \mathbb{c}) \in E_e^d\}$; $\mathbb{C}_e^s \leftarrow \{\mathbb{c} \mid (e, \mathbb{c}) \in E_e^s\}$
16 **for** $\mathbb{c} \in \mathbb{C}_e^d \cup \mathbb{C}_e^s$ **do** $\mathbb{c}.e \leftarrow e$;
17 **for** $\mathbb{c}^s \in \mathbb{C}_e^s$ **do**
18 remove elements about e from M, F and U in $\mathbb{c}^s$
19 apply ranking function $\mathcal{S}$ on $\mathbb{C}_e^s$ and $\mathbb{C}_e^d$ (Section 5)
20 $\mathbb{C}^d \leftarrow \mathbb{C}^d \cup \mathbb{C}_e^d$; $\mathbb{C}^s \leftarrow \mathbb{C}^s \cup \mathbb{C}_e^s$
21 **return** $\mathbb{C}^d$ and $\mathbb{C}^s$

4.1 Critical Attributes and Subject Column

A WebTable attribute $a \in \mathbb{w}.r_h$ is considered *critical* if it provides information that is important for the assessment of the resulting Knowledge Carousel. For example, for the *Winners of Kentucky Derby* carousel in Figure 1(a), the attribute *Winner* (from the original WebTable in Figure 2) is critical since this is what the carousel is about. The attributes *Year* and *Jockey* are also critical because they provide further useful information about winners. A *subject column* is a column associated with a special critical attribute that anchors the resulting carousel (*e.g.*, *Winner*) and it intuitively maps to the carousel members.

To determine whether an attribute is *critical*, we prune away some obvious non-critical attributes. In a WebTable, each attribute in the header row has a set of associated values from the data rows, and attributes with mostly empty values do not provide any real information and should be pruned. We then remove any attribute with more than 50% of its values being empty. We also identify any value in the data rows that is a complete sentence. We assume these values are (long) notes on the table, and because carousels must present information that is concise, they are considered non-critical. We then remove any attribute with more than 50% of its values being sentences.

The remaining attributes of the table $\mathbb{w}$ are assumed to be critical, and they are ranked based on their *criticalness* (Section 4.1.1). Such ranking helps determine which table column is subject (Section 4.1.2) and thus will serve as members, and which attributes will serve as facts for the carousel (Section 4.2).

4.1.1 Criticalness Ranking

The criticalness of an attribute $a_i \in \mathbb{w}.r_h$ is computed based on two scores: (i) *topicality*, *i.e.*, how close a_i is semantically connected to the concept being presented in the table; and (ii) *popularity*, *i.e.*, how often a_i is being requested by users. We measure the topicality by leveraging the metadata of the table ($\mathbb{w}.A$) and the popularity by leveraging the queries associated with the table ($\mathbb{w}.Q$). More formally, we define the criticalness of an attribute a_i as

$$critical(a_i) = \underbrace{\sum_{c \in \mathbb{w}.A} \text{sim}(a_i, c)}_{topicality\ score} + \underbrace{\frac{\sum_{(q,\rho) \in \mathbb{w}.Q} \text{sim}(a_i, q) \times \rho}{\sum_{(\cdot,\rho) \in \mathbb{w}.Q} \rho}}_{popularity\ score} \quad (1)$$

where $\text{sim}(\cdot,\cdot)$ corresponds to the cosine similarity between the *tf-idf* vectors of input strings. The topicality score computes how similar the attribute header is to the various metadata of the table, including table captions, section titles, and surrounding text. It helps identify how close the attribute is to the concept being presented by the table (*e.g.*, *Winner* attribute in Figure 2 is very similar to the table caption, "Kentucky Derby winners"). The popularity score computes how similar the attribute header is to the queries related to the page—weighted by their frequency—which helps understand which parts of the table are queried the most. For example, *Winner* and *Time* attributes in Figure 2 are very popular because users often search for the race winners and their corresponding completion times.

4.1.2 Subject Column

The *subject column* is a table column associated with one of the critical attributes that contains information about the concept being described by the table. For example, in Figure 2, the column under the *Winner* attribute makes for a good subject column because it names all the Kentucky Derby winners, which is the topic of the table. The subject column is particularly important to identify carousel members (Section 4.2) and pivot entities (Section 4.3). Our straightforward method for identifying the subject column of a table, which improves upon previous work [2], entails choosing the critical attribute with highest topicality score (Equation 1).

4.2 Carousel Members and Facts

The carousel members (m_i in Definition 2) are essentially named entities mentioned in the data values of the subject column. The member facts are the highest-ranked remaining critical attributes. For instance, if there is a limit of 3 facts to be presented (Figure 1), the top-3 most critical attributes—with the exception of the subject one—are chosen as facts. The fact values (u_{ik} in Definition 2) correspond to the chosen attributes and are retrieved from the same data row as the member entity. Thus, once we are able to identify the subject column and the critical attributes in the table, generating members and facts for the resulting carousel becomes as simple as (i) applying existing named entity linking techniques [15] on the subject column to extract the carousel members, and (ii) properly formatting the critical attribute values from the table into fact values in the carousel. Finally, if there are multiple named entities being extracted from the same subject column value, we select the one with the highest extraction confidence.

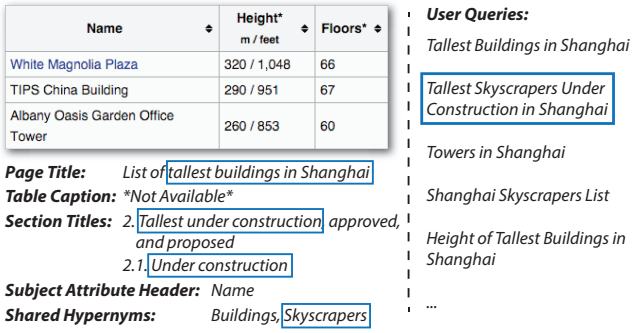


Figure 3: Carousel title generation.

4.3 Pivot Entity

After extracting carousel members and facts, identifying potential pivot entities for each carousel is the next step, which creates an index that associates an entity with multiple carousels. Recall that a pivot entity ($\mathbb{c}.e$) is the entity with which the carousel is associated, *e.g.*, *Winners of Kentucky Derby* (Figure 1(a)) is associated with the pivot entity *Kentucky Derby*. Also recall that downward and sideway carousels define different relationships between the pivot entity and the carousel members, which result in different ways that each type of carousel is generated, as described next. This step of finding pivot entities emphasizes more on recall than precision as the ranking introduced in next section will take care of selecting the best carousels.

4.3.1 Downward Carousel

A downward carousel $\mathbb{c}_e^d$ contains carousel members that collectively form a (set-valued) property of its pivot entity. Intuitively, this means that the underlying WebTable $\mathbb{w}$ that generates the carousel is about the pivot entity and has the property as its subject column. Therefore, a pivot entity for a downward carousel should: (1) not be present in the subject column of $\mathbb{w}$, and (2) be prominent in the context of $\mathbb{w}$, *i.e.*, in $\mathbb{w}.A$ (*e.g.*, "Kentucky Derby" is mentioned in the table caption of Figure 2).

The former condition is easy to deal with by ignoring entities acting as carousel members. The second condition implies $\mathbb{w}.A$ to be the source of pivot entity candidates. Theoretically, many entities can be extracted from $\mathbb{w}.A$ through entity linking, but most of them are unsuitable since the relevance between the carousel and the entity is unknown. Thus, we propose to prune them by requiring a pivot entity being topically coherent to the table context or even the entire web page. Specifically, we identify entities from the table context, count their mentions from the WebTable queries ($\mathbb{w}.Q$), and choose a threshold above which an entity will be considered as a (possible) pivot entity for the table. A Wikipedia page about Kentucky Derby, for example, would rank high in such measure for *Kentucky Derby*, less so for *Kentucky Derby 2015*, and even less so for *Triple Crown Horse Races*: by analyzing user queries, the algorithm discovers *Kentucky Derby* being more frequently mentioned than the other two entities.

We set relatively low thresholds in the above pruning strategy since we want to maximize recall of the pivot entity identification. We leave the task of improving precision in Section 5, which studies the relevance between pivot entity and carousel members.

4.3.2 Sideway Carousel

In comparison, a sideway carousel $\mathbb{c}_e^s$ of a pivot entity contains members that are conceptually similar to the pivot entity, *e.g.*, for the *Triple Crown Horse Races in the USA* in Figure 1(b), the pivot entity *Kentucky Derby* is similar to the members *Preakness Stakes* and *Belmont Stakes* by being all part of Triple Crown races. This makes the identification of pivot entities for sideway carousels much simpler than for downward carousels: all the entities from the subject column of a WebTable are potential pivot entities for the resulting Knowledge Carousel.

4.4 Carousel Title

Generating concise human-readable titles for Knowledge Carousels is of crucial importance. Consider the sideway carousel for *Taylor Swift* shown in Figure 4(d). Without the title, it may not be straightforward to users in what coherent way is *Taylor swift* related to all member entities of the carousel (*U2*, *Roger Waters*, *AC/DC*, etc.). Nevertheless, title generation is a challenging task due to the noisy nature of Web pages, from which WebTables are extracted. Therefore, while there is a number of seemingly promising approaches for title generation, none of them have the desired quality, as described in Section 4.4.1. We have designed our own candidate title generation method by leveraging user queries, which we detail in Section 4.4.2.

4.4.1 Conventional Approaches

Perhaps the most obvious approach would be to use *table captions* as titles. There are, however, two major issues: (i) only 10% of the tables have a table caption (*e.g.*, the table depicted in Figure 3, which lists the tallest skyscrapers under construction in Shanghai, has no caption), and (ii) even when a table caption is present, it does not always describe properly the relationship between the pivot entity and the carousel, in particular for sideway carousels.

The second approach is to use the *page title* as the carousel title. In this case, coverage is not an issue. However, it is often the case that a single Web page has multiple tables with different content: by applying the same title to all the tables, we cannot capture each table's idiosyncrasy. For instance, the Web page that contains the table from Figure 3 has four other tables, and the page title itself does not reflect the fact that all the buildings are still under construction.

Finally, one could use *section titles* or *a combination of page title and section titles*. There are two main issues though: (i) only a small percentage of tables can be properly located within a section, so coverage becomes a challenge, and (ii) combining multiple pieces of text into a concise and yet readable text is by itself a challenging natural language generation problem.

4.4.2 Our Approach: Leveraging User Queries

Given the shortcomings of the aforementioned approaches, we designed a solution that takes advantage of *user queries* associated with the WebTable. Our main intuition is that some user queries can serve as good carousel titles, and the challenge is then how to find them and to assure their quality. Queries are often succinct and understandable to users: they are issued by the users themselves after all. However, there are a few challenges in using them. First, since a single Web page may contain multiple WebTables, some WebTables may be more popular and others may have a sparse query coverage. Second, they are noisy: only a subset of these queries are relevant to the Knowledge Carousel since queries may refer to other content from the Web page from which the carousel was generated. Last, queries may be syntactically incorrect, lacking readability to be a qualified title.

Algorithm 2: Title Generation.

Input: Query set $\mathbb{w}.Q$, table metadata $\mathbb{w}.A$, subject column header with shared hypernyms S'
Output: Best title candidate

```
f ← an empty array
for i ← 1 to |w.Q| do
    if not satisfy hard constraints in Equation 2 then
        continue
    f[i][0] ← DescriptivenessScore(w.A, S', q_i) as in Equation 2
    f[i][1] ← ReadabilityScore(q_i) using language model
index ← FindIndexWithMaxPair(f)
return q_index
```

Our first goal is to increase the query coverage for WebTables, which increases the chance of having good titles. Specifically, we mine structured query templates from search queries, and apply them to the rest of the page to generate more queries. Similar to [1], we can significantly increase the content coverage by using such approach.

Next, to remove noise from the set of queries, we identify the most descriptive user query that can serve as the title for a given carousel $\mathbb{c}$. Specifically, we model the *descriptiveness* of a user query for a carousel based on how it matches the table auxiliary metadata ($\mathbb{w}.A$) and the subject column ($S \in \mathbb{w}.r_h$). For instance, among all the user queries in the right side of Figure 3, the second query covers most of the terms within the table metadata (as highlighted in blue boxes); as a matter of fact, this is the correct title for the table, and it is also substantially superior to the ones that would be obtained using the techniques previously presented. Formally, we adopt the following optimization formula to choose the queries with the best descriptiveness:

$$\arg \max_{(q,\cdot)\in \mathbb{w}.Q} \Big(\sum_{c\in \mathbb{w}.A} \frac{|W_q \cap W_t|}{|W_c|} + \max_{s\in S'} \frac{|W_q \cap W_s|}{|W_s|} \Big) \quad (2)$$
$$s.t. \quad W_q \subseteq (W_A \cup W_{S'}), \quad N_q \subseteq (N_A \cup N_{S'})$$

where W and N denote bag of words and noun phrases [3], respectively, and S' includes the subject column header text S and, additionally, all shared hypernyms[9] of the members of $\mathbb{c}$. This scoring function adds up the similarity between a query and the different components of the table auxiliary metadata as well as the subject column descriptions,[10] thus increasing the comprehensiveness of the carousel title. We also enforce two hard constraints on W_q and N_q, which dictates that a valid query should only contain a subset of words and noun phrases that appear in A and S', indicating that a valid title should be succinct without semantic shift.[11]

[9]Hypernyms are identified by finding sufficient evidence on the Web using Hearst patterns [11].

[10]Since many of the hypernyms are ad-hoc or synonyms, we only give credit to the candidate having the most matched words.

[11]Noun phrases contain multiple words and preserve their order, ensuring that a query only containing "Book" won't be matched to "Book Character."

Since the descriptiveness score only models how a title matches the carousel and ignores the title syntax, the next goal is to measure *readability* to ensure that a qualified user query reads like a title. To achieve this, we train an n-gram language model on the POS tags of web-scale table captions. The model is then able to give a readability score by averaging the probabilities of predicting the current word given the previous words in the candidate title.

Algorithm 2 describes the algorithm for iterating over the template-expanded query set and identifying the best title. After generating the two scores for all the titles, function $FindIndexWithMaxPair$ selects the title candidates having the highest descriptiveness scores. If more than one candidate has the same score, the one with the highest readability score is chosen. Figure 4 depicts some carousels generated by our approach. Note that the titles are concise and human-readable, and they well describe the information being presented, including members and facts.

5. CAROUSEL RANKING

For entities with more than one associated Knowledge Carousel, ranking becomes essential: typically, only one or a few carousels would be shown in a search result page for a query. While table search ranking has been studied before, there are differences from the table search problem as discussed in Section 2, including user intent, features used in table retrieval, and entity-based representation of carousels. Therefore, we believe ranking techniques used for table search are not immediately applicable to carousels, and we have designed our own techniques. In this section, we describe these techniques for scoring the carousels for a given pivot entity for ranking purposes, *i.e.*, designing function $\mathcal{S}$ in Definition 3. We note that the existing ordering of members in the underlying WebTable is often deliberate, and thus we do not reorder the members. The carousel scoring function $\mathcal{S}$ is designed based on *popularity* $\mathcal{P}$ and *relatedness* $\mathcal{R}$.

Popularity ($\mathcal{P}$). The popularity score $\mathcal{P}$ measures how important the underlying WebTable $\mathbb{w}$ of the carousel $\mathbb{c}$ is to users: the more frequently queries lead to $\mathbb{w}$, the more likely $\mathbb{c}$ provide information of interest to users. We use the set of user queries, $\mathbb{w}.Q$, that reside in the page containing $\mathbb{w}$. However, this is a *page-level signal*, since the queries and their clicks are associated with the Web page to which $\mathbb{w}$ belongs, rather than with $\mathbb{w}$ itself. To address this mismatch, we use the table metadata $\mathbb{w}.A$ together with the header information $\mathbb{w}.r_h$ to transform the page-level popularity to the *table-level* popularity using the following equation:

$$\mathcal{P}(\mathbb{c}) = \sum_{(q,\rho)\in\mathbb{w}.Q} \max_{c\in H} \text{sim}(q,c)\times\rho \quad \text{s.t.} \quad H = \mathbb{w}.r_h \cup \mathbb{w}.A \setminus \{\textit{page title}\} \tag{3}$$

where metadata $\mathbb{w}.A$ (except page title) includes contextual information about the table, such as the text surrounding the table and table caption; header information $\mathbb{w}.r_h$ essentially represents the schema of the table; $\text{sim}(\cdot,\cdot)$ corresponds to the cosine similarity between the *tf-idf*-weighted vectorized bag-of-words from the respective input strings.[12] By considering only user queries that match the terms in $\mathbb{w}.A$ or $\mathbb{w}.r_h$, we potentially eliminate user queries that are not relevant to the table. Thus, the resulting popularity measure is a better estimation of the table popularity than the original set of user queries.

[12]Note the resemblance to the criticalness formula in Section 4.1.1.

Relatedness ($\mathcal{R}$). Given that a user queries for entity e, the relatedness $\mathcal{R}$ of a Knowledge Carousel $\mathbb{c}$ measures how related the members of $\mathbb{c}$ are to the pivot entity. Since downward and sideway carousels capture different relationships between the pivot entity and the carousel members, we leverage different relatedness signals for the two types.

For downward carousels, we consider query result co-occurrences [7]. Conceptually, two entities are similar if users querying for one often find the other in the query results. For example, if winning horses are highly relevant to *Kentucky Derby*, a search for *Kentucky Derby* will likely lead to some documents containing these horses. The relatedness score between two entities e_1 and e_2 is computed by counting the occurrences where e_1 appeared in the query and e_2 appeared in the top-ranked documents for that query, and vice versa, normalized by the popularities of both entities. This is aligned with downwards carousels, whose goal is to find entities that are related by the same property or aspect.

For sideway carousels, we consider query refinements [20]. Two entities are considered to be related if users often search for them within the same query session. For example, a user who is interested in Triple Crown races may issue successive queries containing *Kentucky Derby*, *Preakness Stakes*, and *Belmont Stakes* within the same search session. The score between two entities e_1 and e_2 is computed by counting the co-occurrences of e_1 and e_2 in the same user query session and then normalizing by the popularities of both entities.

$\mathcal{R}$ is further divided into two components: *relatedness strength* ($\mathcal{R}_S$) and *relatedness coverage* ($\mathcal{R}_C$). The former is the total sum of the scores for related carousel members; the latter is the fraction of the top-k most related entities to the pivot that are carousel members (we set k to 500). To put all together, let e be the pivot entity of the carousel $\mathbb{c}$, $\text{rel}(e,e')$ be the relatedness score between two entities, and $e.R_k$ be the set of top-k entities related to e. We have:

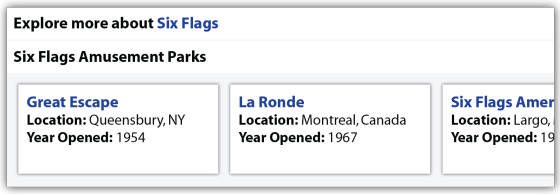
$$\mathcal{R}(\mathbb{c}) = \underbrace{\left(\sum_{m\in\mathbb{c}.M} \text{rel}(e,m)\right)}_{\textit{strength}\,(\mathcal{R}_S)} \times \underbrace{\left(\frac{|\{m|m\in\mathbb{c}.M, m\in e.R_k\}|}{|\mathbb{c}.M|}\right)}_{\textit{coverage}\,(\mathcal{R}_C)} \tag{4}$$

Overall ($\mathcal{S}$). The final ranking function is defined below.

$$\mathcal{S}(\mathbb{c}) = \mathcal{P}(\mathbb{c}) \times \mathcal{R}(\mathbb{c}) \tag{5}$$

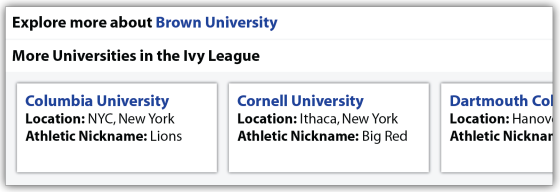
6. EXPERIMENTAL EVALUATION

In this section, we describe the experimental evaluation that we carried out to validate our approach. The goal of the evaluation is the following. First, we assess the entity coverage of our approach (Section 6.1). Second, we examine whether WebTables are effective in generating coherent sets of carousel members (Section 6.2). Third, we evaluate the quality of the carousel ranking and the generated facets for a given query entity (Sections 6.3 and 6.4, respectively). Last, provided that WebTables are a good source for generating carousels and our ranking quality is good, we assess the quality of our title generation solution against alternative approaches (Section 6.5).



(a) Downward carousel for *Six Flags*.

(b) Downward carousel for *Christmas Music*.



(c) Sideway carousel for *Brown University*.

(d) Sideway carousel for *Taylor Swift*.

Figure 4: Knowledge Carousel examples generated by our framework.

Carousel Corpus. The WebTables corpus used in our evaluation has a total size of about 170M tables out of which 8M are from Wikipedia. After running our pipeline over English Wikipedia tables on 200 machines using map-reduce, we obtained around 24.4M sideway carousels distributed over 3.2M pivot entities and 1.6M downward carousels distributed over 516K pivot entities. The pipeline took $O(10)$ hours for both sideways and downwards. The higher number of sideway carousels is expected since more pivot entities are identified for sideways, as any entity from the subject column of a WebTable can potentially serve as a pivot entity.

Pivot Entities. We aimed to perform our evaluation on a typical query workload. Using the most frequent queries is not ideal since it captures many keywords of primarily navigational intent such as *amazon* and *facebook*. Instead, we constructed a workload of the most frequent queries subject to having a diversity of Freebase types (people, TV programs, events, locations, etc.) by limiting the number of queries belonging to any given type to at most 5. The selection process was as follows. We first randomly sampled 10,000 queries from the query log. These queries were piped into an in-house entity recognizer and resolver to ground the entity mentions to Freebase machine IDs. Only the queries containing exactly one entity, with no additional tokens, were kept. Then we scanned through these entities, maintaining counts per type, and rejected any entity that was non-distinct or if the count for its type exceeded the maximum capacity. From the remaining, we sampled 200 entities without replacement.

6.1 Entity Coverage

Methodology. We investigated the coverage of our carousels on *tail* vs *head* entities in our workload after classifying 33 out of 200 of them as tail based on majority vote of 3 raters; Table 2 gives some examples of these. We also examined coverage of a curated feature in Google's Knowledge Panels (KP) that has the flavor of carousels; for instance, given a movie entity, "Action Movies" is sideway-like and "Cast" is downward-like.[13]

[13]We did not compare against the "People also search for" feature from KP because it does not have a notion of sideways or downwards.

Results. Based on the full workload of 200 entities, only 21 yielded a sideway-style result in KP and from among only three verticals — TV shows & movies, video games, and books — whereas our approach produced a result for 138 entities. Downward-style results in KP had more comparable coverage to carousels (64 vs 74 entities, respectively). For tail entities, our approach could generate sideways and downwards for, respectively, 19 and 6 entities; in contrast, only 0 and 2 of the tail ones have sideway- and downward-like KP, respectively. In addition, sideway and downward carousels could be generated for 117 and 67 of the head entities, respectively, while for KP, these numbers are 20 and 40. In Section 6.4, we show that the extra coverage provided by our approach often comes with good quality.

6.2 Quality of Carousel Generation

Methodology. To evaluate how effective WebTables are in providing coherent sets of entities for both downward and sideway carousels, we conducted the following user study. We asked 12 raters (none of whom is an author of this paper) to compare the set of carousel members from the highest ranked carousel (proposed approach) against the set of entities generated by leveraging a Knowledge Base and query logs (baseline approach) as described below. Raters were allowed to use a search engine to better understand pivot entities with which they were not familiar. Each pivot entity was shown along with two sets of member entities,[14] corresponding to the proposed and baseline approaches, as a side-by-side for evaluation, with the left and right sides randomly swapped and formatted identically so that raters would not know which side corresponded to which approach. We only presented entities for which both of the approaches yielded member sets: 85 for sideways and 64 for downwards. The raters were then asked whether *the left side was better*, *the right side was better*, *both sets were decent*, or *both sets were bad*.

Baseline. Though details of entity recommendation methods in existing search engines are proprietary (and, in the

[14]We withheld surrounding metadata, such as header rows or table captions from our approach and property names from the baseline, since they can be misleading when taken out of context and since the metadata from the two sides may not be comparable.

Table 2: Examples of *tail* and *head* entities and their corresponding member descriptions from downward-like Knowledge Panels (KP) and downward carousels ("–" indicates no result found).

Tail Entities	KP	Carousel	Head Entities	KP	Carousel
Attila	Members	Outlawed Album	*Ben Stiller*	Movies and TV shows	Filmography (Actor)
Cutaneous Condition	–	List of HLA alleles associated w/ cutaneous conditions	*CNN*	TV shows	Former Programs
Great Trek	–	Largest First Waves Trek Parties	*FC Barcelona*	Roster	Players
IMI Galil	–	Other Variants of IMI Galil	*Gangnam Style*	Other recordings of this song	Weekly Charts
Houston Marathon	–	Race Winners	*Gwen Stefani*	Songs	Filmography
Middle Tennessee State University	Notable alumni	–	*Inside Out*	Cast	Soundtrack
Neo-Bulk Cargo	–	List of International Auto Shipping Companies	*SpongeBob SquarePants*	Episodes	Episodes

case of Yahoo! Spark, no longer online), we tried to capture the essence of [4, 14] in a baseline method which combines a Knowledge Base with query logs. We used Freebase since its *types* and *properties* naturally correspond to sideways and downwards, respectively. However, we noticed that the number and diversity of types was only rich for head entities (*e.g.*, *Barack Obama*). For example, there are only four types for *Kentucky Derby*. Therefore, we define the notion of a *collection*. A collection of a given entity is the set of entities sharing a property relation from some other entity not in the collection. For example, given entity *Kentucky Derby*, the collection */base/horse_racing/horse_race_of_type* was obtained as a property of the entity *Graded Stakes Race*, which contains *Kentucky Derby* as one of its members and also contains *Preakness Stakes* and *Breeders' Cup Classic*. This technique yields a considerable number of candidate collections: on average, there are 962 collections for a given query entity. We use those collections to generate sideway carousels in the baseline approach.

For generating downward carousels in the baseline approach, we use *property-sets*. A property-set of a query entity is defined to be a set of entities which share the same edge type from the query entity. For example, from *Game of Thrones*, the property *regular_cast/actor* can be used to find all actors of the show. On average, there are 14 properties (with repeated values) for a given query entity.

To rank collections or property-sets and select their respective members from Freebase, we employ the query co-occurrence signal used by the "*People also search for*" feature from Google. The idea behind this is that the co-occurrence signal promotes relatedness while the Freebase collections/properties impose homogeneity. Given a pivot entity e, we generate a sideway set S_{KB} and a downward set D_{KB} as follows:

1. Retrieve the set of entities R from the "*People also search for*" feature for entity e.

2. *Sideway Set*: choose the collection c to which e belongs whose members C are entities e' from triples (o, p, e') for which there also exists triple (o, p, e) with the same property p and have the greatest resemblance to R via Jaccard similarity, *i.e.*:

$$C = \arg\max_{C' \in \mathcal{C}_e} \frac{|R \cap C'|}{|R \cup C'|}$$

   where $\mathcal{C}_e$ is the set of collections to which e belongs. The sideway set S_{KB} is composed of entities belonging to C (minus e).

3. *Downward Set*: choose the property p of e whose members P are objects from triples (e, p, o) of the same property p and have the greatest resemblance to R via Jaccard similarity, *i.e.*:

$$P = \arg\max_{P' \in \mathcal{P}_e} \frac{|R \cap P'|}{|R \cup P'|}$$

   where $\mathcal{P}_e$ is the set of properties of e. The downward set D_{KB} will be composed of entities belonging to P.

Results. The results are shown in Figure 5. For sideways, a majority of the raters preferred the WebTable based approach for 48.24% of the queries and the baseline for 31.76%. Namely, when raters perceived a quality difference, 50% more carousels generated by the proposed approach were deemed better than those generated by the baseline approach. For the remaining cases where there was no quality difference, a majority of raters considered 10 of the 85 carousels (11.76%) to have both sides bad, and both sides were considered decent for the remaining 8.24%. Adding ratios together, our approach yields decent results at least 56.48% of the time.

The results for downwards are similar. The proposed approach beat the baseline 43.75% of the time, versus 29.69% for the baseline approach, another nearly 50% advantage when quality differences were perceived. A majority of raters considered 8 of the 64 pivot entities (12.5%) to have both sides bad, while both sides were considered decent for 14.06%. Combining the ratios, the proposed approach produces reasonable results in at least 57.81% cases.

Upon examining examples where carousels were preferred, it appears that a more tightly-knit set of members was the main reason; *e.g.*, for *Gwen Stefani* (sideways), users preferred to see the presenters of the TV show "The Voice" as members rather than a generic list of female vocalists (baseline). For the cases rated *both bad* for sideways, we noticed that several represented categories and genres rather than specific objects. These do not make for good sideways because they are better suited to being a collection name rather than members of a collection. Examples of such entities and their sideways include *shoe* ("Insulation for Clothing Ensembles") and *broadway theatre* ("Sutton Foster in Theatre"). Another source of problems for the proposed approach was disambiguation of the pivot entity, such as between an eponymous album title and the musician who recorded it (*e.g.*, *Taylor Swift*).

6.3 Quality of Carousel Ranking

Methodology. Our goal here is to demonstrate that the carousel ranking algorithm we developed is choosing good

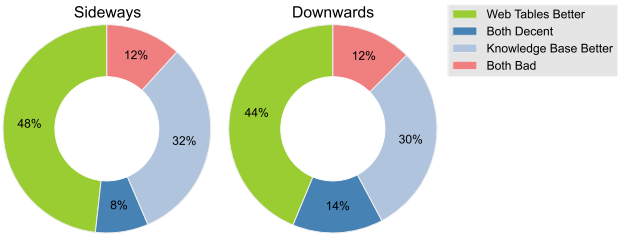


Figure 5: Comparison of WebTable-based approach against baseline.

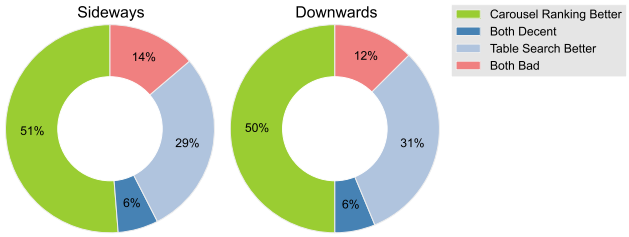


Figure 6: Comparison of top-ranked carousel against Table Search-based one.

carousels from the set of candidates produced by the carousel generation algorithm. Since users have different information needs, there is no single best ranking of carousels that would satisfy all users. Our goal, thus, is to provide carousels that many users (or perhaps the "average" user) would appreciate. Unfortunately, having a large pool of raters order carousel candidates by preference for each pivot entity to establish a "gold standard" ranking is too costly (in time and money) and has the usual pitfalls of full rankings when the items being ranked are not amenable to simple ranking functions [9]. Therefore, we use pairwise comparisons to demonstrate that users prefer carousels that are high-ranked by our ranking algorithm to those that are high-ranked by the baseline approach (and low-ranked by us) as detailed below.

We presented to raters (the same ones from the study in Section 6.2) a side-by-side comparison of the top-ranked carousel against an alternatively-ranked one as determined by the state-of-art table search engine, Google's Table Search [2]. Table Search was used for comparison because it ranks WebTables based on query keywords and, therefore, provides a "good" carousel (*i.e.*, one for which it can be reasonably expected to rank high). For each pivot entity and for each carousel type, among the same pool of carousel candidates generated by our carousel generation algorithm, users were asked to compare the highest ranked carousel by our ranking algorithm against the one belonging to the table that Table Search determined to be the highest-ranked table. It is worth noting that, while Table Search has no concept of downwards and sideways, it does tend to favor tables of downward nature to the pivot query. Furthermore, in the rare case that none of the Table Search results coincided with a carousel, we dropped the query from the evaluation. We also discarded the query when the top result from Table Search was the same as our top-ranked carousel: 5 for sideways, 48 for downwards. The high drop-out number for downwards is expected because of the tendency for Table Search to find property tables of the pivot entity query. This resulted in a remainder of 80 sideways and 16 downwards for the ranking evaluation.

Results. The results are shown in Figure 6. For sideways, our top-ranked carousel was considered to be better than the Table Search-based carousel 51.25% of the time while the latter was preferred 28.75% of the time. This is a nearly 80% advantage for our ranking algorithm when quality differences were perceived by the raters. A majority of raters considered both carousels to be bad in 13.75% of the cases, and in the remaining 6.25%, both sides were considered decent with no preference. These ratios were very similar for downwards, where our carousel ranking algorithm obtains a 60% advantage over Table Search where it is preferred by raters 50% of the time versus 31% for Table Search.

6.4 Quality of Facets

Methodology. Here we evaluate if the concept representing carousel members ("facets") are decent. In the evaluation from Section 6.3, raters were shown member sets in context, enabling them to understand the underlying concept. Hence, we assume carousel results for cases with "both decent" imply reasonable facets and those with "both bad" do not. Furthermore, we assume that cases where a carousel was chosen over Table Search imply a decent result. We asked 4 raters (different from Section 6.3) to examine the cases where Table Search was chosen over sideway carousels to see if and how many of these carousels had decent quality facets and counted only those for which at least 3 chose "yes".

Results. Half (11 out of 22) of the carousels were considered decent. Summing these plus the cases where a carousel was preferred or both sides were decent gave a total of 57 cases, *i.e.*, 71% of the sideway carousels had decent facets. From the sideways for tail entities, 6 of 8 facets were considered decent. One example of a bad facet was "College Athletic Programs in Tennessee" for *Middle Tennessee State University* as opposed to "List of Colleges and Universities in Tennessee", which was chosen by Table Search. Here, all the metadata based on user queries is not very rich and includes mostly sports-related entities. For downwards, the ratio of entities with decent facets was higher but the coverage was much lower for tail entities. We believe this is due to lack of table content in Wikipedia. For example, while *Cargo* has carousels, the (more "drilled down") tail entity *Neo-bulk cargo* does not.

6.5 Quality of Carousel Titles

Methodology. To assess the quality of the carousel titles, we performed the following evaluation. For each pivot entity, we presented the set of members from our top-ranked carousel along with a multiple choice of candidate titles. Six raters were asked to select which title(s) best explains the set, with multiple choices allowed for cases of a tie. In addition to our generated title from the techniques proposed in Section 4.4, there were up to three candidate titles used as a baseline that were chosen from the Web page containing the originating WebTable, namely, the page title, section title, and table caption, if available. Also, raters could choose *none* if they disliked all of the candidates. The candidate titles in the multiple choice were randomly permuted to make it difficult to determine to which method they corresponded. In total, this evaluation included 60 each for sideways and downwards.

Results. The results are displayed in Figure 7. For 14 sideways and 8 downwards, a majority of raters disliked all titles and chose *none*. There were 30 sideways and 34 downwards for which the generated title was preferred versus 16 and 20,

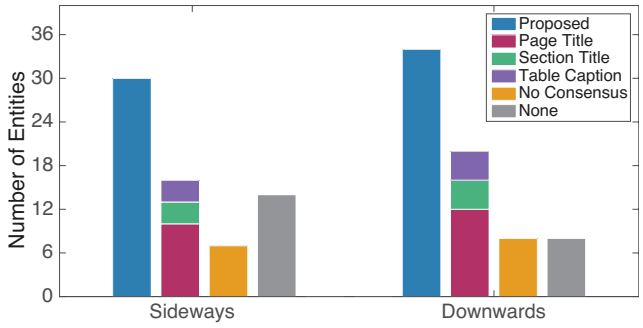


Figure 7: Comparison of proposed title generation method against baselines.

respectively, for one of the baseline candidates. Note that we grouped all baselines together as a single histogram bar with colors breaking down the various alternatives: for sideways, 10 from page title, 3 from section title, and 3 from table caption; for downwards, 12 from page titles, 4 from section title, and 4 from table caption. This is being generous to the baseline since it assumes an oracle has chosen the best from the alternatives. In addition, there were 7 sideways and 8 downwards for which there was no majority vote, some of which may provide additional good titles.

Upon examining the titles considered to be of poor quality, we noticed that the tables tended to exist on pages containing many other tables and within nested subsections of the page, for which it was more difficult to attribute user queries to tables. An example for downwards is "Chart Performance of Jingle Bell Rock" for a table specifically pertaining to the recording by Bobby Helms. We believe that more complicated templates can bring improvement.

7. CONCLUSION

In this paper, we presented a study on leveraging the WebTable corpus to provide Knowledge Exploration via two types of queries, *sideways* and *downwards*. We discussed various technical challenges in generating high quality Knowledge Carousels from tables on the Web and proposed solutions, including how to: (i) select pivot entities for the two carousel types; (ii) create a coherent and meaningful set of entity members for the carousels; (iii) generate a human-readable carousel title that is concise, interesting, and consistent with the set of members; and (iv) rank carousels based on popularity and relatedness. Our experimental evaluation shows that WebTables are a good source of Knowledge Carousels, generating sets of members that provide meaningful and interesting information to users. Using WebTables, we are also able to generate complete and concise titles, and provide a ranking function that, overall, gives relevance to interesting carousels. To the best of our knowledge, our paper is a first study on enabling coherent Knowledge Exploration using data sources other than Knowledge Bases.

As future work, we plan to run our approach with other tables on the Web. We will also study different machine learning techniques to construct ranking models and compare with our ranking function. Our current approach does not take into account user-specific query signals, and we plan to include these to develop personalized carousel rankings.

8. REFERENCES

[1] G. Agarwal, G. Kabra, and K. C.-C. Chang. Towards Rich Query Interpretation: Walking Back and Forth for Mining Query Templates. In *WWW*, 2010.

[2] S. Balakrishnan, A. Y. Halevy, B. Harb, H. Lee, J. Madhavan, A. Rostamizadeh, W. Shen, K. Wilder, F. Wu, and C. Yu. Applying WebTables in Practice. In *CIDR*, 2015.

[3] S. Bergsma and Q. I. Wang. Learning Noun Phrase Query Segmentation. In *EMNLP*, 2007.

[4] R. Blanco, B. B. Cambazoglu, P. Mika, and N. Torzec. Entity Recommendations in Web Search. In *ICSW*, 2013.

[5] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: Exploring the Power of Tables on the Web. *VLDB*, 1(1):538–549, 2008.

[6] N. Craswell and M. Szummer. Random Walks on the Click Graph. In *SIGIR*, 2007.

[7] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic Query Expansion Using Query Logs. In *WWW*, 2002.

[8] B. B. Dalvi, W. W. Cohen, and J. Callan. WebSets: Extracting Sets of Entities from the Web Using Unsupervised Information Extraction. In *WSDM*, 2012.

[9] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank Aggregation Methods for the Web. In *WWW*, 2001.

[10] R. Gupta, A. Halevy, X. Wang, S. E. Whang, and F. Wu. Biperpedia: An ontology for Search Applications. *VLDB*, 7(7):505–516, 2014.

[11] M. A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *CCL*, 1992.

[12] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and Searching Web Tables Using Entities, Types and Relationships. *VLDB*, 3(1-2):1338–1347, 2010.

[13] T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active Objects: Actions for Entity-centric Search. In *WWW*, 2012.

[14] I. Miliaraki, R. Blanco, and M. Lalmas. From "Selena Gomez" to "Marlon Brando": Understanding Explorative Entity Search. In *WWW*, 2015.

[15] D. Milne and I. H. Witten. Learning to Link with Wikipedia. In *CIKM*, 2008.

[16] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Exemplar Queries: Give Me an Example of What You Need. *VLDB*, 7(5):365–376, 2014.

[17] R. Pimplikar and S. Sarawagi. Answering Table Queries on the Web Using Column Keywords. *VLDB*, 5(10):908–919, 2012.

[18] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc Object Retrieval in the Web of Data. In *WWW*, 2010.

[19] R. Qian. Understand Your World with Bing. *Official Bing Blog, March*, 2013.

[20] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering Query Refinements by User Intent. In *WWW*, 2010.

[21] A. Singhal. Introducing the Knowledge Graph: Things, not Strings. *Official Google Blog, May*, 2012.

[22] A. Thalhammer, M. Knuth, and H. Sack. Evaluating Entity Summarization using a Game-based Ground Truth. In *ICSW*, 2012.

[23] C. Wang, K. Chakrabarti, Y. He, K. Ganjam, Z. Chen, and P. A. Bernstein. Concept Expansion Using Web Tables. In *WWW*, 2015.

[24] F. Wu, J. Madhavan, and A. Halevy. Identifying Aspects for Web-search Queries. *Journal of Artificial Intelligence Research*, 40(1):677–700, 2011.

[25] M. Yang, B. Ding, S. Chaudhuri, and K. Chakrabarti. Finding Patterns in a Knowledge Base using Keywords to Compose Table Answers. *VLDB*, 7(14), 2014.

[26] X. Yu, H. Ma, B.-J. P. Hsu, and J. Han. On Building Entity Recommender Systems using User Click Log and Freebase Knowledge. In *WSDM*, 2014.