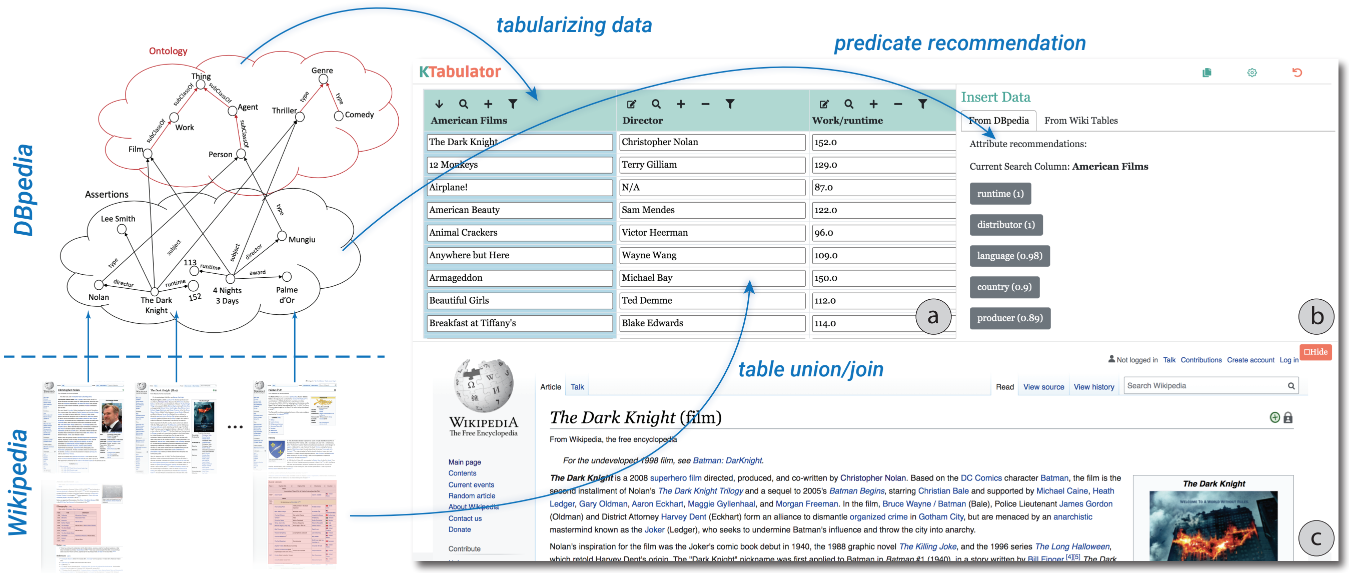
KTabulator: Interactive Ad hoc Table Creation using Knowledge Graphs

Siyuan Xia
University of Waterloo
Waterloo, ON
s9xia@uwaterloo.ca

Nafisa Anzum
University of Waterloo
Waterloo, ON
nanzum@uwaterloo.ca

Semih Salihoglu
University of Waterloo
Waterloo, ON
semih.salihoglu@uwaterloo.ca

Jian Zhao
University of Waterloo
Waterloo, ON
jianzhao@uwaterloo.ca



American Films	Director	Work/runtime
The Dark Knight	Christopher Nolan	152.0
12 Monkeys	Terry Gilliam	129.0
Airplane!	N/A	87.0
American Beauty	Sam Mendes	122.0
Animal Crackers	Victor Heerman	96.0
Anywhere but Here	Wayne Wang	109.0
Armageddon	Michael Bay	150.0
Beautiful Girls	Ted Demme	112.0
Breakfast at Tiffany's	Blake Edwards	114.0

Attribute recommendations:

Current Search Column: **American Films**

runtime (1)
distributor (1)
language (0.98)
country (0.9)
producer (0.89)

The Dark Knight (film)

From Wikipedia, the free encyclopedia

For the undeveloped 1998 film, see Batman: DarKnight.

The Dark Knight is a 2008 superhero film directed, produced, and co-written by Christopher Nolan. Based on the DC Comics character Batman, the film is the second installment of Nolan's *The Dark Knight Trilogy* and a sequel to 2005's *Batman Begins*, starring Christian Bale and supported by Michael Caine, Heath Ledger, Gary Oldman, Aaron Eckhart, Maggie Gyllenhaal, and Morgan Freeman. In the film, Bruce Wayne / Batman (Bale), Police Lieutenant James Gordon (Oldman) and District Attorney Harvey Dent (Eckhart) form an alliance to dismantle organized crime in Gotham City, but are menaced by an anarchistic mastermind known as the Joker (Ledger), who seeks to undermine Batman's influence and throw the city into anarchy.

Nolan's inspiration for the film was the Joker's comic book debut in 1940, the 1988 graphic novel *The Killing Joke*, and the 1996 series *The Long Halloween*, which retold Harvey Dent's origin. The "Dark Knight" nickname was first applied to Batman in *Batman* #1 (1940), in a story written by Bill Finger.[4][5] *The Dark*

Figure 1: KTabulator: an interactive system to help users effectively extract, build, or extend ad hoc tables from large corpora (e.g., Wikipedia), by leveraging their computerized structures in the form of knowledge graphs (e.g., DBpedia). KTabulator supports effective information seeking by extracting and suggesting relevant entities, properties, and tables. The main Interface of KTabulator consists of: (a) Table Panel showing the current state of the table being created, (b) Data Action Panel containing two tabs where users can insert data from DBPedia or tables in Wikipedia, and (c) Wikipedia Panel allowing users to browse the Wikipedia pages of selected entities in the table.

ABSTRACT

The need to find or construct tables arises routinely to accomplish many tasks in everyday life, as a table is a common format for organizing data. However, when relevant data is found on the web, it is often scattered across multiple tables on different web pages, requiring tedious manual searching and copy-pasting to collect data. We propose *KTabulator*, an interactive system to effectively extract, build, or extend ad hoc tables from large corpora, by leveraging their computerized structures in the form of knowledge graphs. We developed and evaluated KTabulator using Wikipedia and its knowledge graph DBpedia as our testbed. Starting from an entity or an existing table, KTabulator allows users to extend their tables by finding relevant entities, their properties, and other relevant tables, while providing meaningful suggestions and guidance. The results of a user study indicate the usefulness and efficiency of KTabulator in ad hoc table creation.

CHI '21, May 8–13, 2021, Yokohama, Japan
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445227

CCS CONCEPTS

• **Information systems** → **Information integration**; • **Human-centered computing** → **Interactive systems and tools**.

KEYWORDS

Data tables, data cleaning, data integration, database, user interface

ACM Reference Format:

Siyuan Xia, Nafisa Anzum, Semih Salihoglu, and Jian Zhao. 2021. KTabulator: Interactive Ad hoc Table Creation using Knowledge Graphs. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3411764.3445227

1 INTRODUCTION

A table is a natural and commonly used data format to organize and share data that captures facts about sets of entities and their properties. Once data is in a tabular format, using simple operations such as filtering, sorting, or aggregating columns, humans can explore and analyze it very efficiently. The need to find or construct tables arises routinely to accomplish many tasks in every day life. Consider searching for books that were recently written by Asian developmental economists, a student seeking information about public engineering universities across provinces, or a basketball enthusiast analyzing past draft picks of NBA teams, among many other examples. Many of these tasks are highly varied and ad hoc, which makes it very challenging for people to find readily-available tables on the web that directly satisfy their needs. When relevant tables are found, they are often incomplete and existing systems do not provide an easy way to complete the missing information according to the needs of users. As a result, preparing the tables can be a tedious task, requiring manual copy-pasting from browsed information, or searching for related tables that can be unioned or joined with existing ones using a spreadsheet software.

At the same time, obtaining specific information about individual entities on the web has become easier over time. There is vast unstructured textual information on the web, such as in online encyclopedia, news websites, and academic articles. They include factual information on a wide range of entities, such as politicians and celebrities, entertainment productions and events, and geographical or scientific entities. The information in some of these corpora have lately been computerized as structured information in the form of *knowledge graphs*. Perhaps the most popular of these, such as DBpedia [11], YAGO [43], Wikidata [48], and Freebase [14], have been constructed by computerizing online encyclopedia, such as Wikipedia [7] or Baidu Baike [5]. Data in these knowledge graphs are stored in some graph-structured data model, often as a set of *Resource Description Framework* (RDF) triples [1], which makes it suitable for being processed by software. These triples have facilitated several applications, such as question answering in search engines: e.g., the answer to the question *Who is the director of The Dark Night?* can be encoded and found in the triple: (`dbr:The_Dark_Knight`, `dbo:director`, `dbr:Christopher_Nolan`) [2] in DBpedia.

In this paper, we aim to help users efficiently create ad hoc tables they need by leveraging a knowledge graph. We contribute an interactive and exploratory data gathering system called *KTabulator* (Figure 1) that uses Wikipedia and its associated knowledge graph DBpedia as a test-bed for this purpose. In contrast to existing applications in which a software, such as a search engine, processes a knowledge graph, *KTabulator puts the power of knowledge graphs directly into the hands of humans.* That is, in a human-in-the-loop manner, users start creating tables from an entity or a pre-existing table in a Wikipedia article and interactively transform these tables by: (i) adding sets of related entities as *new rows*; or (ii) adding new properties of existing entities in these tables in *new columns*. Similar to the initial step, additional entities and properties can be extracted directly from DBpedia as well as pre-existing tables in the Wikipedia articles.

KTabulator primarily aims to empower non-technical users by streamlining iterative (1) data gathering, and (2) table transformation steps when creating ad hoc tables. KTabulator's main interface is designed to be familiar to users who are creating tables and support column and row additions and guide them as they explore and gather possibly sparse, heterogeneous and inconsistent data from both DBpedia and Wikipedia tables. Specifically, we have adopted a familiar spreadsheet interface to support table creation operations. To provide guidance for data exploration, KTabulator keeps a mapping from each entity in the created table to the corresponding nodes in DBpedia. The neighborhoods of these nodes are used in several ways to guide the user in selecting further transformations. For example, if a column contains a set of movies, users can add existing properties of the nodes in DBpedia that correspond to these movies, such as runtimes, release dates, or directors as new columns. Similarly, if a user populates a new director column, the system uses the DBpedia ontology to understand that directors are persons (`dbo:Person`) and suggests other properties of movies that are persons, such as writers or cinematographers. The system also explores Wikipedia pages of these neighbors to search for pre-existing Wikipedia tables that the user might union or join with their tables. To address the sparsity and inconsistencies in the data in DBPedia and Wikipedia tables, KTabulator allows heterogeneity in the sets of entities and properties that are added to users' tables.

There exist classes of systems that partially provide KTabulator's functionalities. As users interact with KTabulator and gather data from DBPedia, it automatically generates SPARQL queries. Similar functionality is provided by systems that aim to make it easier for users to query and explore knowledge graphs through interactive query building or natural question answering [10, 21, 47, 54]. Similarly, KTabulator provides interactive table transformation operations that are provided by spreadsheets and systems for data cleaning and transformation [3, 9, 28, 32, 37]. KTabulator brings together functions from these different classes of systems as a solution to address the problem of ad-hoc table creation.

To validate KTabulator, we conducted a user study with 12 participants on table creation tasks, including a definitive task with a target table and a exploratory task around a predetermined topic. We report our findings in quantitative task performance, subjective questionnaire ratings, and qualitative feedback. Participants appreciated the capabilities of KTabulator in efficiently adding rows or columns directly extracted from Wikipedia/DBpedia, with the the help of system automation as well as suitable guidance and suggestions. The results also indicate that KTabulator is effective and useful for supporting ad hoc table creation in a human-in-the-loop approach, which allows participants to easily collect the information they needed.

2 BACKGROUND

We provide a brief background on DBpedia, knowledge graphs, RDF, and SPARQL and refer the readers to references [1, 4, 6, 11], for more details on these technologies. DBpedia is a knowledge graph, i.e., a set of RDF triples, that are extracted from Wikipedia pages [11]. Each RDF triple is of the form (subject, predicate, object) and expresses a fact about an entity or a property of an entity that is covered in Wikipedia. Subjects, predicates, and objects are uniquely identified with *universal resource identifiers* (URIs). For example in DBpedia, the URI http://dbpedia.org/resource/The_Dark_Knight, abbreviated as `dbr:The_Dark_Knight`, identifies the movie The Dark Knight. There are two types of triples in DBpedia:

- *Ontological triples:* These express the metadata (or schema information) about entities. These include triples with predicates `rdfs:domain`, `rdfs:range`, `rdf:type`, `rdfs:subPropertyOf`, and `rdfs:subClassOf`, among others. For example, (`dbo:director`, `rdfs:range`, `dbo:Person`) indicates that directors should be of type `dbo:Person`. KTabulator uses ontological triples in its recommendation algorithms that suggest relevant data to users.
- *Assertion triples:* These are the remaining triples that assert facts about entities, such as (`dbr:The_Dark_Knight`, `dbo:director`, `dbr:Christopher_Nolan`). Users gather and tabulate these assertion triples to construct tables.

When subjects and objects are viewed as nodes, triples can be seen as labeled edges, and one can view a set of triples as forming a graph (hence the term knowledge graph). The ontological subset of knowledge graphs is sometimes referred to as an ontology.

The standard query language to query data that is stored as a set of RDF triples is SPARQL [4]. SPARQL is a declarative query language that is similar in structure to SQL, and consists of a WHERE clause (corresponding to FROM in SQL) that expresses joins in the form of graph patterns, a FILTER clause (corresponding to WHERE in SQL) to express predicates over the matched patterns, and a SELECT clause, as in SQL, to return a set of matched variables that were used in the WHERE clause. For example, the following "star" query:

```
SELECT ?a
WHERE ?a rdf:director dbr:Christopher_Nolan ,
      ?a dct:subject dbc:crime_thriller
```

returns all nodes that have an `rdf:director` edge pointing to `dbr:Christopher_Nolan` and another `dct:subject` edge pointing to `dbc:crime_thriller`, which visually forms a star shape.

3 RELATED WORK

There has been several work on designing interfaces and tools for various applications that use knowledge graphs and other semantic web technologies, such as RDF and SPARQL, that focus on various topics related to human-computer interactions, such as efficient ways of visualizing, browsing, or searching in knowledge graphs or authoring ontologies to increase humans' accessibility to these technologies. We refer the reader to reference [18] for a good survey of the papers that cover HCI-related topics. Below, we cover several of the work that are closer to our work.

Smart Assistance in Spreadsheets: Closest to our work is the *SmartTable* [50] spreadsheet system which suggests new rows and column headers in a spreadsheet application. This work assumes a setting where a user is manually creating a table cell by cell in a spreadsheet application and is in the middle of entering a new cell. Similar to an auto-completion system, SmartTable ranks a list of candidate values for the cells or new column headers. The algorithms for suggesting a cell value or column headers are based on an earlier work by the same authors [51]. These algorithms take as input the cell values, table caption, and column headers the user has entered into the spreadsheet and use data from DBpedia and Wikipedia tables to suggest either the cell value or column header the user is about to edit. In contrast, KTabulator assumes a different setting in which users insert sets of entities or their properties and entire columns instead of individual cells. For example, KTabulator supports adding sets of entities that satisfy a set of properties from DBpedia or unioning users table with other tables, which avoids moving from cell to cell to populate tables. Similar to SmartTables, KTabulator also supports finding relevant data from DBpedia, but it is designed for users to directly browse, explore, and query DBpedia and Wikipedia. Instead, SmartTables allows an interaction with DBpedia and Wikipedia data only through suggestions.

Systems for Data Extraction and Simplifying Querying: There are several query building systems to simplify querying of databases that store knowledge graphs. We can divide these systems into two broad categories: i) direct database querying systems; and ii) question answering systems. Systems for database querying provide interactive frameworks to define a query using visual illustrations. These systems expose the database structure and schema to the users and enable users to browse the schema and interactively formulate queries [21, 26, 41, 42, 47]. For example, Sparklis [21] allows users to formulate queries of DBPedia using faceted search where users interactively click drop down menus and text boxes to formulate queries, which are also shown in a natural language. The results of queries are then presented in several formats such as flat or nested tables. On the other hand, question answering tools translate questions posed in natural language to SPARQL to retrieve results from a knowledge graph [8, 10, 17, 54]. All of these tools help non-tech users to extract data from knowledge bases. As such they can provide part of the data gathering functionality of KTabulator. However, these systems do not provide functionality to interactively expand, union, and join query results in iterations. Instead they assume that the query or question results that are returned, even if they are in a table format, are final and further query or question results override previous results. Users in KTabulator instead iteratively build tables that can contain results of multiple queries.

Data Visualization Using Knowledge Graphs: ViDAX [20], LinkDaViz [45], and Filter Dials [22] are systems that are designed for users to visualize the data in knowledge graphs in aggregate using different visualizations. Similar to our work, ViDaX uses DBpedia as a data source and initially displays the ontology of DBpedia as a radial tree to users. Then users can click on different types in the ontology, e.g., `dbo:Person` and then several numeric and date predicates and can create several charts that show aggregates of some other predicates, e.g., users can generate a 2D visualization of the number of people that were born in different years by picking `dbo:birthYear` predicate and count aggregation. Filter Dials is system that supports a specialized visualization called *filter dials*, that presents counts of number of entities with different sets of predicates. LinkDaViz targets a more general use case where a user first browses and selects a set of entities and predicates from the ontology of a knowledge graph. Then LinkDaViz's visualization suggestion framework suggests a list of visualizations, such as bar or line charts to display the selected data. Similar to KTabulator, users in these systems interactively query a knowledge graph but for the purpose of data visualization instead of ad hoc table creation.

Browsing, Exploring, and Visualizing Knowledge Graphs: Tabulator [12] is a browser for browsing the web and semantic web using various views, such as a map or a calendar view that can interpret and display location or date-related RDF tags on pages. Tabulator allows primarily the browsing of RDF triples in the semantic web but users can also access the original web sources that these triples annotate. Several other work have proposed systems, such as /facet [25] and tFacet [15], that supports *faceted exploration* of knowledge graphs. Faceted exploration is a browsing technique that organizes content into orthogonal categories. A survey on these techniques and systems can be found in reference [46]. Prior work has also developed systems, such as WebVOWL [30], LODmilla [31], LODlive [16], RelFinder [24], and Semantic Data Explorer [35], which visualize entities and the relationships among the entities from a knowledge graph in the form of graphs. These systems focus on browsing, exploring, and visualizing knowledge graphs and their sources. Although browsing DBpedia and Wikipedia is supported in KTabulator by clicking different cells, KTabulator's main interface is a spreadsheet interface and its main goal is to facilitate table creation.

Query Answering With Tables and Table Union Search: An interesting line of work in information retrieval is answering user queries with tables or columns or cells of existing tables [29, 36, 44, 52, 53]. Often these work assume a user asks a keyword query in a natural language, such as "Mountains in North America," and a search system returns as results a part of a table. These systems do not support ad hoc table creation. They help find relevant but often incomplete tables to the needs of users but do not provide a mechanism to complete the missing information through exploratory iterative extensions as KTabulator.

Several recent work has investigated the problem of finding joinable and unionable tables [33, 55] to a given input table. This problem is a specific case of a core problem in data integration called *schema matching* [40]. These references assume the *linked data* setting where a user wants to extend a publicly available dataset T, say published for the public by a government, with other publicly available published datasets in the web. Some of these techniques have also been integrated into a search system called Toronto Open Dataset Search [56] that allows users to navigate open datasets from the web by finding joinable tables. Finding joinable and unionable tables is also related to the task of entity completion, which focuses on finding a set of related entities to a starting set of seed entities [23, 49]. These works focus on designing core search algorithms instead of providing interactive capabilities for users to explore data and generate tables from web data.

Data Transformation Systems: Researches from both HCI and database communities developed different interactive systems for transforming datasets from one structure to another [3, 9, 28, 32, 37]. Wrangler [28] offers an interactive framework to create tables from unclean unstructured data. The tool relies on a predictive framework and guides users throughout the data cleaning and transforming process by providing suggestions on the following steps. GraphWrangler [9] is a system to interactively create graphs out of relational tables. Ultrawrap Mapper [38] also creates graphs out of relational databases but the system is not interactive and performs an automatic mapping between datasets. KTabulator also transforms data between structures but instead transforms data in a knowledge graph into a table (in addition to transforming tables into extended tables). These data transformation systems assume that the data has already been gathered. Another data transformation and cleaning tool, OpenRefine [3], provides some limited capabilities of enriching data from other sources (e.g., Freebase), however, users still need to have their own data to start with and the tool does not provide functionalities like integrating existing tables or guidance for data exploration. KTabulator brings together both data transformation and gathering functionalities.

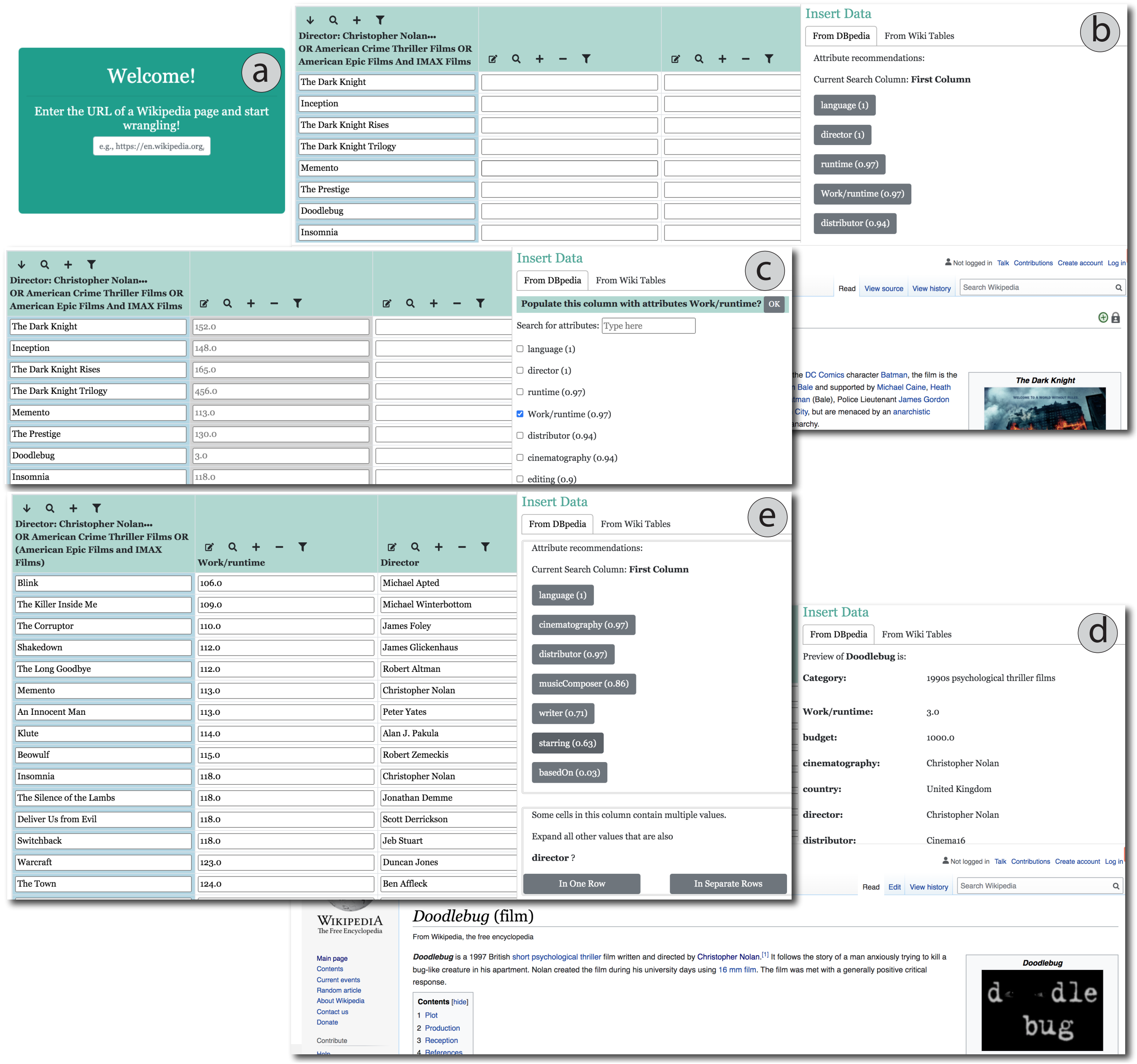
4 USAGE SCENARIO

In this section, we demonstrate a scenario of how KTabulator can help users easily and quickly create an ad hoc table. To gain a better understanding, we recommend you to watch the accompanying video in our supplementary materials.

Ivan, a movie enthusiast, wants to prepare a list of around ten movies for Karissa to watch at home, during the COVID-19 period. Instead of simply choosing from the top rated movies from IMDB, Ivan wants to ensure that the list of movies suits Karissa's interest the best. He decides to use KTabulator for this task.

Starting by creating a primary column. Ivan starts by copying and pasting the Wikipedia URL of Karissa's favourite movie *The Dark Knight* onto KTabulator's landing page (Figure 2a). He then chooses the action "Create a table about The Dark Knight" indicated by the system. KTabulator then suggests a list of properties that Ivan can use to populate the primary column of the table. He selects the property `director: Christopher Nolan` to add all movies directed by Nolan, whose other movies Ivan suspects Karissa might also like. This adds a total of only 12 movies. To broaden his options, Ivan reflects on what Karissa likes about The Dark Knight: it is a crime thriller movie with excellent visuals, and adds two more sets of movies to the table by clicking ↓ on the header of the primary column: (i) movies sharing the Wikipedia category `American crime thriller films`; and (ii) movies with both the `IMAX films` category and the `American epic films` category (creating a customized category of `IMAX American epic films`) (Figure 2b).

Expanding tables by adding and manipulating columns. Ivan next decides to add more information about the movies in his table to help him filter some of them out. He clicks on ✎ of an empty column header, which updates the right panel, as shown in Figure 2c, with a list of properties of the entities in the *search column*, which is by default the primary column. KTabulator extracts these properties by exploring in DBpedia the neighborhood of the nodes that represent the movies in the primary column. For each property, the panel also shows the fraction of entities in the search column that have the property. Ivan sees and clicks on the `Work/runtime` property, which 97% of the movies in the search column has, and



Welcome!

Enter the URL of a Wikipedia page and start wrangling!

e.g., https://en.wikipedia.org,

Director: Christopher Nolan... OR American Crime Thriller Films OR American Epic Films And IMAX Films		
The Dark Knight		
Inception		
The Dark Knight Rises		
The Dark Knight Trilogy		
Memento		
The Prestige		
Doodlebug		
Insomnia		

Insert Data

From DBpedia From Wiki Tables

Attribute recommendations:

Current Search Column: **First Column**

language (1)

director (1)

runtime (0.97)

Work/runtime (0.97)

distributor (0.94)

Director: Christopher Nolan... OR American Crime Thriller Films OR American Epic Films And IMAX Films		
The Dark Knight	152.0	
Inception	148.0	
The Dark Knight Rises	165.0	
The Dark Knight Trilogy	456.0	
Memento	113.0	
The Prestige	130.0	
Doodlebug	3.0	
Insomnia	118.0	

Insert Data

From DBpedia From Wiki Tables

Populate this column with attributes Work/runtime? OK

Search for attributes: Type here

- [ ] language (1)
- [ ] director (1)
- [ ] runtime (0.97)
- [x] Work/runtime (0.97)
- [ ] distributor (0.94)
- [ ] cinematography (0.94)
- [ ] editing (0.9)

Not logged in Talk Contributions Create account Log in

Read View source View history Search Wikipedia

the DC Comics character Batman, the film is the Bale and supported by Michael Caine, Heath man (Bale), Police Lieutenant James Gordon City, but are menaced by an anarchistic anarchy.

The Dark Knight

Director: Christopher Nolan... OR American Crime Thriller Films OR (American Epic Films and IMAX Films)	Work/runtime	Director
Blink	106.0	Michael Apted
The Killer Inside Me	109.0	Michael Winterbottom
The Corruptor	110.0	James Foley
Shakedown	112.0	James Glickenhaus
The Long Goodbye	112.0	Robert Altman
Memento	113.0	Christopher Nolan
An Innocent Man	113.0	Peter Yates
Klute	114.0	Alan J. Pakula
Beowulf	115.0	Robert Zemeckis
Insomnia	118.0	Christopher Nolan
The Silence of the Lambs	118.0	Jonathan Demme
Deliver Us from Evil	118.0	Scott Derrickson
Switchback	118.0	Jeb Stuart
Warcraft	123.0	Duncan Jones
The Town	124.0	Ben Affleck

Insert Data

From DBpedia From Wiki Tables

Attribute recommendations:

Current Search Column: **First Column**

language (1)

cinematography (0.97)

distributor (0.97)

musicComposer (0.86)

writer (0.71)

starring (0.63)

basedOn (0.03)

Some cells in this column contain multiple values.

Expand all other values that are also

director ?

In One Row In Separate Rows

Insert Data

From DBpedia From Wiki Tables

Preview of **Doodlebug** is:

Category:	1990s psychological thriller films
Work/runtime:	3.0
budget:	1000.0
cinematography:	Christopher Nolan
country:	United Kingdom
director:	Christopher Nolan
distributor:	Cinema16

Not logged in Talk Contributions Create account Log in

Read Edit View history Search Wikipedia

WIKIPEDIA The Free Encyclopedia

Main page Contents Current events Random article About Wikipedia Contact us Donate

Contribute

Doodlebug (film)

From Wikipedia, the free encyclopedia

Doodlebug is a 1997 British short psychological thriller film written and directed by Christopher Nolan.[1] It follows the story of a man anxiously trying to kill a bug-like creature in his apartment. Nolan created the film during his university days using 16 mm film. The film was met with a generally positive critical response.

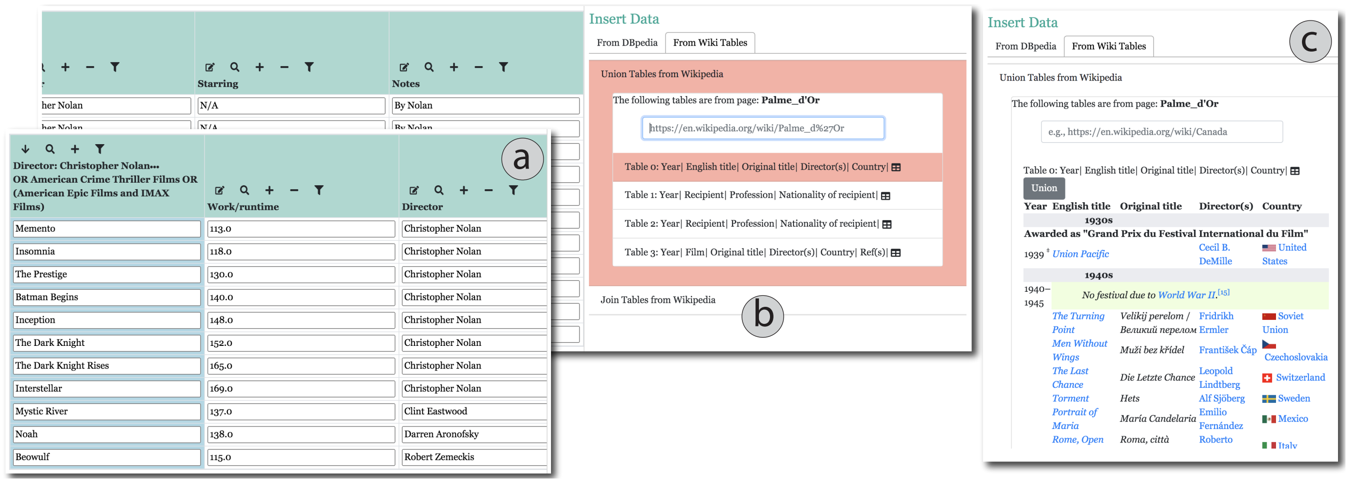
Contents [hide]
1 Plot
2 Production
3 Reception
4 References

Doodlebug

Figure 2: Ivan is using KTabulator to prepare a list of movies for Karissa to watch at home. (a) Pasting the Wikipedia URL of *The Dark Knight* to the landing page. (b) Populating the primary column with movies directed by `Christopher Nolan` and those under `American crime thriller films` and IMAX `American epic films`. (c) Adding a movie *work/runtime* column. (d) Browsing the Wikipedia page of *Doodlebug*. (e) Choosing from the properties recommended by KTabulator.

populates the new column with this property (Figure 2c). Ivan decides that his recommendations to Karissa should neither be too short (less than 1 hour and 45 minutes) nor too long (longer than 3 hours). To select such movies, he first sorts his table according to ascending runtime order, using the "Sort Ascending" option under ▼. Before filtering out the movies, Ivan notices something odd about the runtimes: two movies, *Doodlebug* and *Quay*, have really short runtimes, 3 and 8 minutes, respectively.

Exploring entities using DBpedia and Wikipedia Panels. Ivan is curious who these movies are directed by and adds a new column to the table and from the list of properties picks `director`. He notices that both of these short movies are surprisingly directed by Nolan. To explore these movies more, he double-clicks on the



Director: Christopher Nolan••• OR American Crime Thriller Films OR (American Epic Films and IMAX Films)	Work/runtime	Director
Memento	113.0	Christopher Nolan
Insomnia	118.0	Christopher Nolan
The Prestige	130.0	Christopher Nolan
Batman Begins	140.0	Christopher Nolan
Inception	148.0	Christopher Nolan
The Dark Knight	152.0	Christopher Nolan
The Dark Knight Rises	165.0	Christopher Nolan
Interstellar	169.0	Christopher Nolan
Mystic River	137.0	Clint Eastwood
Noah	138.0	Darren Aronofsky
Beowulf	115.0	Robert Zemeckis

Starring	Notes
N/A	By Nolan
N/A	By Nolan

Insert Data

From DBpedia | From Wiki Tables

Union Tables from Wikipedia

The following tables are from page: **Palme_d'Or**

https://en.wikipedia.org/wiki/Palme_d%27Or

Table 0: Year| English title| Original title| Director(s)| Country|

Table 1: Year| Recipient| Profession| Nationality of recipient|

Table 2: Year| Recipient| Profession| Nationality of recipient|

Table 3: Year| Film| Original title| Director(s)| Country| Ref(s)|

Join Tables from Wikipedia

Insert Data

From DBpedia | From Wiki Tables

Union Tables from Wikipedia

The following tables are from page: **Palme_d'Or**

e.g., https://en.wikipedia.org/wiki/Canada

Table 0: Year| English title| Original title| Director(s)| Country|

Union

Year	English title	Original title	Director(s)	Country
	1930s			
Awarded as "Grand Prix du Festival International du Film"				
1939 ‡	*Union Pacific*		Cecil B. DeMille	United States
	1940s			
1940–1945	*No festival due to World War II.*[15]			
	The Turning Point	*Velikij perelom / Великий перелом*	Fridrikh Ermler	Soviet Union
	Men Without Wings	*Muži bez křídel*	František Čáp	Czechoslovakia
	The Last Chance	*Die Letzte Chance*	Leopold Lindtberg	Switzerland
	Torment	*Hets*	Alf Sjöberg	Sweden
	Portrait of Maria	*María Candelaria*	Emilio Fernández	Mexico
	Rome, Open	*Roma, città*	Roberto	Italy

Figure 3: Karissa is using KTabulator to refine the list of movies from Ivan. (a) Receiving the list of movies created by Ivan. (b) Discovering tables to union on the Wikpedia page of "Palme d'Or." (c) Selecting the table to union.

table cell for one of the two movies, *Doodlebug*, which updates the right panel and the bottom Wikipedia panel of KTabulator (Figure 2d). He reads the updated Wikipedia pages of the movies and surprisingly learns that Nolan has directed two short movies, a thriller from his college years and a recent documentary from 2015. However, he does not think Karissa would be interested in these movies and proceeds to remove the too long and too short movies by using the "Filter" option of the runtime column using .

Exploring entities through property suggestions. Ivan next notices KTabulator's suggestions on the right panel (Figure 2e) to add other properties related to the director property that he just added. These include `writer`, `cinematography`, `musicComposer`, `starring`, among others. KTabulator uses the semantic information from DBpedia's ontology to make this suggestion (specifically all of these properties have a range of type person, i.e., their `rdfs:range` is `dbo:Person`). Ivan next decides to add movies with Karissa's favourite actors, so clicks on `starring` to add a new column. Since most movies feature multiple actors and actresses, Ivan chooses to show all the different actors for each movie in a single merged cell (an option offered by KTabulator), instead of the other option of flattening the cell into multiple rows. He finds Karissa's favourite actors Anthony Hopkins and Sean Penn star in several movies, such as *Noah* and *Mystic River* and adds the "Favorite Movie Star!" note in a custom "Notes" column for these movies.

Sharing tables. Ivan decides to keep some of the remaining Nolan movies in his list and adds "By Nolan" as a note on these rows. Finally, he filters out all movies with an empty "Notes" column and shares the result table with Karissa.

Enhancing tables using existing tables in Wikipedia. Karissa indeed finds many interesting movies in Ivan's table (henceforth *the extracted table*) as shown in Figure 3a. However, after adding the "Country" attribute and seeing most of the movies from the extracted table are from the United States, she decides to make the list more international. She decides to add some of the Palme d'Or winners from the Cannes festival. She notices that she cannot find award-related properties of the movies when she tries to add a new set of movies directly in KTabulator (because the DBpedia entries of the movies do not contain award information). Instead, she searches for "Palme d'Or" in the Wikipedia panel (Figure 3b) and locates a pre-existing Award-winners table with 5 columns: `Year`, `English title`, `Original title`, `Director(s)`, and `Country`. She clicks on "Union Table" action from the "Table Actions" tab in the right action preview panel (Figure 3c), and the system suggests ways to align several of the columns in the extracted table with those in the Award-winners table. Karissa accepts the suggested alignment `First Column`↔`English title`, `director`↔`Director(s)`, and `country`↔`Country`. KTabulator adds the rows of the Award-winners table to the extracted table according to the alignment and automatically populates the `starring` column for the new movies from the Palme d'Or Award-winners table using the information from DBpedia.

5 DESIGN GOALS

Four major design goals guided our design of KTabulator. These goals were informed by existing principles for efficiently creating and manipulating tables [27, 28, 37] existing properties of knowledge graphs [39, 50, 56, 57], and through several design iterations.

G1: Support union and join operations for data insertion. Tables consist of a set of rows (or tuples) which are records about entities, their properties, and their relations to other entities. Therefore, in order for users to create tables efficiently, we needed to support two set-oriented operations that insert data to users' tables in bulk from DBpedia: (i) *union*, which extends a table with a new set of rows; and (ii) *join*, which extends the records in a table with new columns. For example, users in KTabulator indicate one or more criteria, such as "movies directed by Nolan", and insert as new rows into their tables the set of entities in DBpedia that satisfy the criteria. This contrasts with a design that would allows record-by-record data insertion, say possibly as the user browses DBpedia entries of individual entities. Here we had a choice to limit the support of these operations only on the knowledge graph DBpedia, which is the main source of structured information we focus on. However, we observed that there are many high-quality pre-existing tables in Wikipedia articles that can supplement missing information in DBpedia. Therefore we decided to support unioning and joining users' tables also with pre-existing tables from Wikipedia.

G2: Provide guidance and suggestions during table creation. Our second design goal was based on the observation that information in Wikipedia and DBpedia is very large. For example, there can be hundreds of properties and thousands of triples in DBpedia even for a single entity and very large and multiple tables on single Wikipedia pages. For example, there are over 2000 triples about the city Berlin in DBpedia and over 100 tables across 74 Wikipedia pages about NBA drafts since 1947. Since, manually searching, browsing and finding relevant information at this scale cannot be efficient, KTabulator needs to provide guidance and suggestions to users. As we explain in Section 6, KTabulator heavily utilizes the *semantic information* that is stored as ontological triples in DBpedia about entities, e.g., triples with predicates `rdf:type` or `rdfs:range`, to suggest relevant entity properties that can be added as new columns or finding relevant Wikipedia pages that contain relevant pre-existing tables.

G3: Embrace data heterogeneity, inconsistencies, and sparsity. Our third design goal was that the system should assume that the data in the table will be heterogeneous and inconsistent, and sometimes sparse. This is based on several observations. First, the set of entities that users add to their tables often satisfy different sets of properties. For example, a user might want to create a table of Prime Ministers of the UK and the presidents of China. Second, because Wikipedia pages are created by humans, the extracted data in DBpedia have inconsistencies, e.g., similar properties of entities can be tagged with different RDF predicates. Therefore, the system should be flexible enough to keep heterogeneous sets of entities and properties in the table. The data in DBpedia is also very sparse, i.e., many entities also have many missing values. KTabulator presents the sparsity levels of properties of entities to users (to avoid unnecessarily adding too sparse columns) and bases its attribute suggestion rankings taking sparsity levels into account.

G4: Support for Browsing, Processing, and Sharing Information. Table creation is an exploratory and iterative process in nature, so in addition to data insertion functionalities, the system also needs to support basic functionalities to browse, process, and share information presented in the table. Following our dual knowledge graph-original source (DBpedia-Wikipedia) design, KTabulator offers two preview panels to allow users to browse both DBpedia and Wikipedia pages. We also added basic spreadsheet processing functionality, such as sorting, filtering, searching, adding annotations in custom columns, projecting out columns, and exporting and sharing tables.

6 KTABULATOR

6.1 Overview and Main User Interface

Figure 4 shows the overall architecture of KTabulator. KTabulator is a web application with two major components: (i) a browser-based user interface that facilitates table creation and manipulation actions; and (ii) a backend server that is responsible for fetching the data the user has asked from DBpedia and pre-existing Wikipedia tables. The system uses the public DBpedia SPARQL access point at

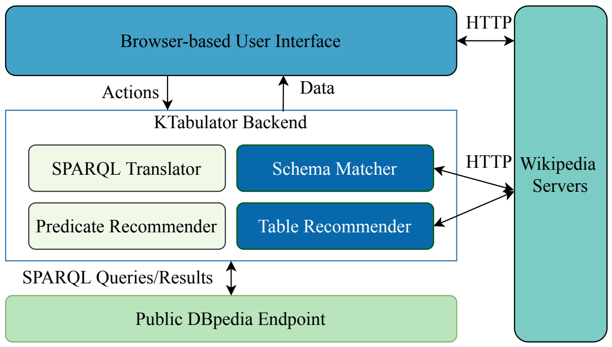


Figure 4: KTabulator architecture. All four components from the backend uses the Public DBpedia Endpoint. Schema Matcher and Table Recommender (drawn in color blue) also communicates with the Wikipedia servers.

http://dbpedia.org/sparql which stores DBPedia triples in OpenLink Virtuoso. To read existing tables in Wikipedia, KTabulator directly reads Wikipedia pages and scrapes the retrieved HTML.

Figure 1 shows the main interface of KTabulator, which consists of three panels. The *Table Panel* is located on the left side and shows the table the user is creating in a spreadsheet view. There are two types of columns: (i) *primary column* is the left most column and is the main set of entities the table contains; and (ii) *secondary columns* are properties of the entities in the primary column or other secondary columns. The values in secondary columns themselves can be entities, e.g., directors of movies, or primitive values, e.g., the release dates of movies. KTabulator keeps a mapping from the entities in the cells of the table panel to the DBpedia URIs and Wikipedia URLs of the entities. The other two panels are the *Data Actions Panel*, which contains two tabs, using which users can insert data into their tables from DBpedia or tables in Wikipedia. Finally, at the bottom is a Wikipedia Panel, which users use to browse the Wikipedia pages of the entities in their tables.

6.2 Table Initiation and Column Type Trees

Users start creating a table by copy-pasting the URL of a Wikipedia page about an entity e. There are two ways to create a table: (i) by clicking "Create a table about e", users can add the entity e as the first cell of the primary column; or (ii) if the Wikipedia page about e contains tables, users can select "Start from an existing table on page" and select one of the tables R from the page, which copy pastes the content of R to the Table Panel. When R is copied over, by default the left most column of R that contains entities (inferred by the anchors in the cells in R) becomes the primary column.

At any point in time, KTabulator maintains a *semantic type tree* s_i for each column c_i in the created table. s_i is the type hierarchy of the entities in c_i that are annotated with the fraction of entities in c_i that have each type in the hierarchy. It is computed as follows. For each e_i in c_i, KTabulator inspects the ontological triples with predicate `rdf:type`. We focus on triples where the object in these triples, i.e. the actual types, are from DBpedia Ontology, i.e., those with prefix `dbo:`. In DBpedia, the `rdf:type` predicate of each entity e_i contains the most specific type t of e_i and recursively the ancestor types of t

(defined through `subClassOf` predicates). For example, the type hierarchy of The Dark Knight is `dbo:Film`→`dbo:Work`→`owl:Thing`. Every entity's type descends from `owl:Thing` in the DBpedia ontology. The union of the hierarchy of each entity forms the type tree of the column (often a simple chain). Then for each type t in this tree, we count the fraction of entities that are of type t, which gives us an annotated tree, where the root `owl:Thing` is annotated with 1.0. As we explain in Section 6.4, the type trees are used when joining or unioning users' tables with existing Wikipedia tables.

6.3 Data Gathering From DBpedia

6.3.1 Adding New Columns (Join) and Column Provenance. Users add a new column to their tables by first selecting one of the existing primary or secondary columns as the *search column* by clicking the 🔍 icon and then clicking on the ✎ icon of an empty column. The KTabulator backend retrieves the triples where the subject is an entity in the search column and displays the union of the predicates in these triples in the Insert Data Panel as in Figure 2(b)[1].

Predicates in DBpedia (and in other knowledge graphs) can be quite sparse and there are often inconsistencies in the predicates. To help users handle data sparsity in DBpedia entries, next to each predicate, KTabulator shows a value between 0 and 1 indicating the fraction of entities in the search column that have the predicate and ranks the predicates in terms of these sparsity levels. The system also allows selecting multiple predicates to add to a column. For example, selecting `dbo:releaseDate` and `dbo:releaseYear` will fill the cells in the column with the union of these predicates. If a user selects multiple predicates and an entity contains multiple triples with these predicate, users can flatten the row into multiple rows or keep a single merged cell with multiple values.

KTabulator stores the predicates $p_1, ..., p_k$ and the search column c_s that were used in populating each column c_i as the *provenance* of c_i. The provenance is used to retrieve these predicates automatically when new rows are added to the table (explained momentarily).

6.3.2 Adding New Rows (Union). Users add new rows to their tables by clicking the ↓ icon on the primary column. As shown in Figure 2a, this displays the union of the predicate-object pairs of subjects that correspond to the entities in the primary column. Wikipedia categories which are object value of `dct:subject` are displayed without `dct:subject` to simplify the interface (e.g., IMAX films in Figure 2a). Users then select any number of the predicate-object pairs from this list and retrieve the set of subjects from DBpedia that have these predicate-object values. For example, selecting `dbo:distributor: Warner Bros` and `American movies` will return all American movies distributed by Warner Bros from DBpedia and added as new rows. To retrieve these movies,the SPARQL Translator at the backend issues a star query with the selected predicate and object pairs to the DBpedia access point. Once the new entities have been added as new rows, existing columns of the rows are automatically filled by retrieving from DBpedia the predicates that form the provenance of each column.

[1]Since knowledge graphs and RDF triples are mainly designed for computers to process, KTabulator shortens the URIs of predicates when displaying to users to make them more readable, e.g., `dbo:writer` and `dbo:musicComposer` are displayed as `writer` and `musicComposer`.

6.3.3 Predicate Recommender (PR). KTabulator adopts the predictive interaction framework to help users find relevant sets of predicates when adding new columns. When the user indicates the intention to add a secondary column, KTabulator's PR module suggests a list of predicates to add as a column as shown in Figure 2d. For the first secondary column, the top 5 highest density predicates are suggested. For an additional column c_n, PR suggests predicates using the following procedure. Suppose the last secondary column added prior to c_n is c_p and c_p's provenance is $p_1, ..., p_k$ of entities in column c_s. PR first computes two sets of predicates P_{sem}, for *sem*antic, and P_{str}, for *string*, as follows. PR retrieves the ontological triple (p_i, `rdfs:range`, t_i), where t_i is the type of objects with predicate p_i and puts these types into a set T. Then PR retrieves the ontological triples (p_i, `rdfs:subPropertyOf`, t_i') and also adds t_i' to T'. Then PR searches DBpedia for other predicates of the entities in c_s that have a `rdfs:range` in T and `rdfs:subPropertyOf` in T'. To compute P_{str}, PR searches for predicates of entities in c_s that have string values similar to $p_1, ..., p_k$ checking for string containment of p_i in other predicates of entities in c_s and vice versa. Finally, PR merges P_{sem} and P_{str} and then ranks these predicates as suggestions in decreasing sparsity levels.

6.4 Data Gathering From Wikipedia Tables

Users in KTabulator can also join and union their tables with existing tables from Wikipedia. The dual data insertion from DBpedia-Wikipedia can happen in any order, e.g., tables that were initially created from DBpedia or Wikipedia tables can be extended further with DBpedia or Wikipedia tables. Users indicate the Wikipedia table R they would like to join or union with the table U they are creating by going to the Wikipedia page that contains R manually, as in our usage scenario (Figure 2e), or by selecting R from the recommendations of the system (explained momentarily).

6.4.1 Join. Both join and union operations with R require the system to match the columns of R and U. This is the schema matching problem [40] in relational data management and is handled by the *Schema Matcher* (SM) module of KTabulator. For joins, SM uses a purely value-based technique. Specifically, for each column r_j of R, SM uses the URLs of the entities in r_i to map them to nodes in DBpedia. Then for each column pair u_i and r_j, SM computes a joinability metric that is computed as the fraction of entities in u_i that are also in r_j (so u_i and u_j would successfully join on these entities). We currently limit joins of R and U to a single column and KTabulator suggests the three highest score column pairs as the join columns and the user confirms one of these pairs or manually indicates another pair.

6.4.2 Union. In contrast to joining, for taking the union of U with R, KTabulator adopts a semantic technique. A value-based technique does not work for matching column for unioning because we expect the values in columns to not overlap when two tables are being unioned. Recall that KTabulator already maintains the type tree of each column of U. Using the URLs in the cells of R, SM also constructs the type tree of the columns of R. The table union algorithm works as follows. For each column u_i in U, SM first checks if there is any column r_j in R with the similar column names (string containment). If yes, these two columns are matched. For each unmatched column r_j in R and u_i in U, SM computes a similarity score $sim(u_i, r_j) = \max_t level_t \times \min\{fr(t, u_i), fr(t, r_j)\}$, where (i) t is all possible types that appear in the type tree of u_i and r_j; (ii) $level_t$ is the level of t where `owl:Thing` is level 0, its child nodes, e..g., `dbo:Work` are level 1, its grandchild nodes, e.g., `dbo:Film` are level 2, so on and so forth. We multiply with level number to prioritize matches on more specific types. Finally, (iii) $fr(t, u_i)$ and $fr(t, r_j)$ are the fraction of entities in u_i and r_j respectively that have type t (recall these are stored in the type tree). Given this similarity metric, we do a greedy mapping where we match the most similar columns u_i and r_j first, then remove those, and then match the second highest, so and so forth. The user can modify the suggested mapping or accept it and KTabulator will add the rows to U by aligning the matched columns of T with their matches in U and project out the other columns of T. SM uses a simple and purely semantic information-based algorithm but more advanced techniques [33, 40] can easily be integrated into KTabulator.

6.4.3 Table Recommender (TR). TR is the module in KTabulator that searches Wikipedia for unionable and joinable tables. If the user started creating a table from a Wikipedia page P, TR searches up to two degree neighbors $nbrs$ of P in DBpedia and then scrapes the Wikipedia pages of these neighbors to find relevant tables. We limited our search to two-degree neighbors because often tables with exact schemas occur in pages of recurring events, such as NBA drafts pages for different seasons, which have distance two in DBpedia. The found tables are ranked according to their unionability score in the "From Wikipedia" tab in Insert Data panel. The unionability score of a table R with user's table U is the number of columns SM was able to match between R and U.

6.5 Other Processing Operations

For general usability, KTabulator supports several other operations: standard spreadsheet operations, such as filtering, sorting, and projecting out columns, exporting the data in csv format, sharing a created table with another user for facilitating collaboration in KTabulator, and browsing Wikipedia and DBpedia entries of the entities in a table by clicking on different cell.

7 USER STUDY

We conducted a user study to assess the effectiveness and usefulness of KTabulator. The general purposes of this study was to investigate how people use the system to create ad hoc tables that fulfill their goals, and understand the strengths and weaknesses of the system.

7.1 Participants and Apparatus

We recruited 12 participants (eight males and four females) via multiple mailing lists at a local university. Two participants are between age 18–24, nine between age 25–34, and one between 35–44. Participants were all graduate students (eight Masters and four PhDs) whose technical backgrounds include computer science, engineering, and biology. Their self-reported familiarity with table manipulation software (e.g., Excel) had a median of 7 and a mode of 7 on a scale of 1 to 7 (1: no familiarity; 7: frequently using such software). We conducted the study via a remote video conferencing software. The system was deployed as a web application and participants accessed it from their personal computers.

7.2 Tasks and Design

As discussed in Sections 1 and 3, two classes of existing tools partially provide KTabulator's functions: tools that aim to make it easier for users to query and explore knowledge graphs, and tools for data cleaning and transformation. A combination of tools from these two classes would enable our use cases and form a baseline, but switching between tools would require copy-pasting and would not be a fair comparison to KTabulator. The closest to our work is SmartTables [50]. However, it is not publicly available and only supports cell-by-cell table manipulation, so cannot support efficient ad hoc table creation. Thus, we decided not to include a baseline in our study design.

To evaluate KTabulator, we designed two tasks for our study:

T1: **Definitive task.** Participants needed to create a specific table with some requirements, starting from a Wikipedia page. In particular, given a Wikipedia page about a university, participants were asked to create a table containing *all the Canadian public universities that have faculty size larger than 100 and are located in cities with population larger than 50,000.* This task requires participants to employ some core features of KTabulator, such as populating entities in the first column, adding additional columns about the entities, and changing the search column.

T2: **Exploratory task.** Participants were given an exploratory goal to create a table containing *a list of 10–15 movies of interest to recommend for their friends.* They could start from any existing page or table by searching on Wikipedia.

T1 has a target table (i.e., the answer) to create, which allows us to validate if participants could quickly learn the features of KTabulator and achieve a meaningful goal. T2 is open-ended, which allows us to examine how participants could equip themselves with KTabulator to answer their own questions. By having these two tasks, the study could help us examine the basic features of KTabulator as well as explore the potential of the system.

7.3 Pilot Study

To test out our study design, we conducted a pilot study with three participants (two males and one female). All participants are between age 25-34, whose backgrounds include computer science, architecture, and tourism. The pilot study started with a pre-study questionnaire, followed by a training session introducing KTabulator's features with pre-recorded videos. Then, participants completed two tasks sessions (i.e., T1 and T2), and filled in a post-study questionnaire, followed by a semi-structured interview. Of the three participants, only one completed the tasks and the other two were stuck especially on T2. This was because participants had difficulty remembering some functions introduced in the training session at the beginning, which were essential for completing tasks (e.g., changing the primary column). We realized that having a single training session covering all of KTabulator's features overwhelmed the participants.

7.4 Procedure

Based on the findings in our pilot study, we adjusted the procedure for our actual study. Specifically, we split the training session (after the pre-study questionnaire) into two. For the first training session, we presented KTabulator with a pre-recorded video introducing the core features of the system, such as adding columns and rows, filtering. Participants were then instructed, step by step, to create a table very similar to that in training video: a list of US presidents who are also Nobel Prize Laureates. The experimenter could answer any questions raised by participants. The above procedure assured that they had the adequate skills and knowledge to complete T1.

Participants were then asked to perform T1; however, they could ask questions related to UI if they were confused. More specifically, six participants asked how to perform a UI action, e.g., how to change the search column. We told them the action, e.g., a button on the UI. Two of these six participants asked an additional question: "Why do I only see one entry in the first column?" Both had selected a very selective predicate, e.g., `facultySize=2457`, when extending their first entity. We asked them to study the column header and both noticed and removed this predicate. Other than these, we did not give any direction or hint to complete the task. After finishing the task, participants filled in the NASA TLX questionnaire based on their experience in T1.

Next, a second training session was conducted, in which another tutorial video was presented, showing some advanced features of KTabulator, such as finding similar tables and unioning tables. Again, the experimenter helped them familiarize with the additional features by creating a table similar to the one in the video: augmenting a table of a Premier League season on Wikipedia.

Then, participants were given up to 15 minutes to complete T2 independently and without back and forths with the experimenter. Similarly, they filled in the NASA TLX questionnaire based on their experience in T2, as well as an exit-questionnaire to gather their impression of KTabulator in both tasks. Finally, we conducted a semi-structured interview to collect their feedback. In the end, participants received $25 for their time and effort. The whole study lasted about 70 minutes for each participant. We screen-captured the task sessions and audio-recorded the interviews.

8 RESULTS

In this section, we report our results from the user study, including both quantitative measures and qualitative feedback. We denote the participant as *P#* in the following.

8.1 Task Performance

On average, participants spent 7min 53s (σ = 4min 6s) for the definitive task (T1). Of all the participants, eight perfectly completed the task by meeting all the requirements, and three partially met the requirements by incorrectly setting the filters for the faculty size or population. P4 did not catch the `country=Canada` condition from the task description "Canadian public universities" and thus ended with a very different table. However, the mistakes due to misinterpretation were less of an indicator of the effectiveness of KTabulator.

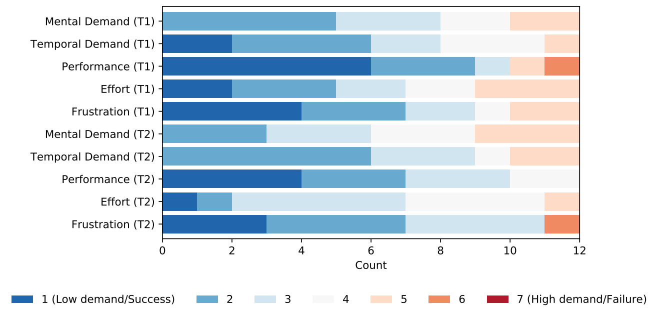
An ideal solution to T1 would be: (1) populate the primary column by universities with `country=Canada` and `type=public`, (2) add `faculty size` and `city`, (3) change the `city` as the search column and add `population`, and 4) filter the `faculty size` and `population`. Of the 11 participants who completed the task well, seven followed a very similar approach, while four added the `country` or the university `type` as a separate column later and used filtering to meet the requirements. This reveals the flexibility of KTabulator, supporting multiple ways to achieve the same goal. Actually, P2 and P7, who missed the condition `country=Canada` initially for the primary column, got back on track by adding `country` as a predicate from KTabulator's predicate recommendation (and later filtering rows with value Canada). In addition, seven participants were able to add `city` and `population` using the predicate recommender, while others manually selected the predicates for each column. Interestingly, P2 used the cell preview for `city`, `population`, and `density`, before manually selecting the `population`. It is encouraging that the guidance and suggestions offered by KTabulator allow most participants to be effective in completing the task, even if they sometimes went off the "optimal" path. During the process, P7 and P11 needed a reminder for changing the search column to `city` in order to add `population`. Currently, KTabulator's suggestions are based on a single search column, which can possibly be extended to all columns in the future.

For the exploratory task (T2), participants were able to create a variate of tables, such as Steven Speilberg's films with awards, 2000's comedy films with starring and director's birthplace, Tom Hank's movies with reasonable length and high budget, etc. The average time taken by participants was 11min 33s (σ = 5min 22s). Seven of 12 participants started their exploration from a Wikipedia page about an entity (e.g., director, movie), and the rest of them started from an existing table. Further, six participants explored predicates one hop away in the knowledge graph from the entity they started, and three explored two hops away, such as adding a column for birthdates of directors of movies in the primary column. Four participants frequently used the predicate recommendation feature to select new predicates to add as a column; specifically, P5 and P10 avoided adding predicates with very low sparsity levels (e.g., `release date`).

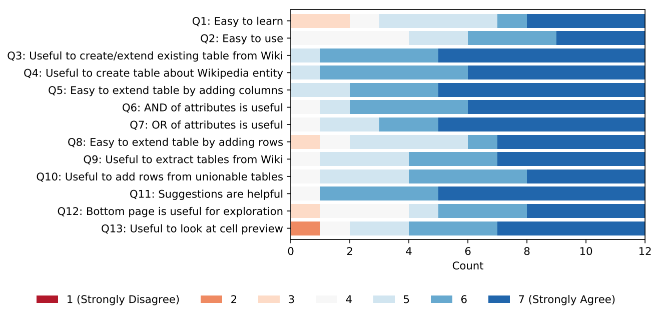
8.2 Questionnaire Ratings

Figure 5 shows participants' ratings on the NASA TLX questionnaires and the exit-questionnaire. We can see that for the definitive task (T1), most participants (at least 9 out of 12) rated 4 or below on each question, indicating that they were comfortable of using KTabulator and felt successful for doing their tasks. P6 rated 6 for the Performance factor, because they only partially completed the task, due to incorrect use of filter. The results of the exploratory task (T2) seem slightly better compared to that of T1, which might be because participants got more familiar with the system or they were more engaged with an open-ended task. Also, for each question, at least 9 out of 12 participants gave a rating of 4 or less. One participant (P14), who tried to find a unionable table in T2, rated 6 on the Frustration factor. This participant started with a Marvel character table and tried to find a table with movie titles to union with the character table. The system did not find unionable tables in this case.

Moreover, the results of the exit-questionnaire, in general, indicate that participants had a good experience of using KTabulator. While two participants gave a score of 3 on "easy to learn," a majority of them thought the system was both easy to learn and use (Q1 and Q2). Participants especially appreciate KTabulator's abilities to



(a)



(b)

Figure 5: (a) Participants' ratings on the NASA TLX questionnaire for T1 and T2 (the lower the better). (b) Participants' ratings on the exit-questionnaire (the higher the better).

create or extend tables from the information on a Wikipedia page (Q3 and Q4). They believed that extracting tables from Wikipedia helps them find information needed for table creation (Q9 and Q10), and the logical AND and OR operations (Q6 and Q7) were helpful for manipulating predicates to add to the tables. The AND and OR operations refer to the ability to select multiple predicates when adding new rows or columns, respectively from DBpedia (recall Section 6). Furthermore, they thought that adding rows and columns was generally easy (Q5 and Q8). A majority of participants applauded for the predicate recommendation feature (Q11). Similarly, they appreciated the Wikipedia page view and the cell preview features (Q12 and Q13); however, for each of the two questions, there was one participant who thought it was not that useful. Their feedback will be reported in the following section.

8.3 Qualitative Results

We conducted thematic analysis on our interviews with the participants. In the following, we report the results in correspondence to the design goals in Section 5 and describe participants' feedback on comparing KTabulator with existing relevant tools.

8.3.1 Data Extraction and Curation (G1 and G3). Supporting data extraction and curation based on knowledge graphs while embracing the specific characteristics of real-world data is an essential part of our design. In general, participants had a good impression to KTabulator, mentioning many potential applications such as creating ad hoc tables for personal interests and recreation, academic research, and any information sharing or comparison tasks. For example, P11 said that *"There are different algorithms with different time complexity and space complexity. It's good to have those in one page using the tool, so you can call it like cheat sheets."*

What participants appreciated the most was KTabulator's ability of saving much time and effort for them to collect relevant information. *"I like the idea that I can aggregate information from multiple Wikipedia pages. Otherwise, I'll probably open a bunch of different tabs and try to figure out information from different pages."*-P1. *"You know that Wikipedia has a bunch of information subjected to the topic. So if you were manually to understand those information without the tool, it will take a lot of time."*-P4. This was echoed by P11: *"You do not have to manually enter the information or write the code to extract the information,"* and P10: *"I can create my own table and curate it the way I want. Suppose I'm trying to find car accidents from the year 2010, I am able to extract that easily if Wikipedia has the information."* But several participants were concerned about the slow performance of KTabulator for operating on large tables.

Moreover, participants were impressed by the powerful join and union operations offered by KTabulator. *"The most useful feature I would say is I can have any one example of something, and it can give me a whole list of that something."*-P5. Similarly, they liked the ability to *"keep on adding different attributes [predicates] to whatever we want to find"*-P6. P12 echoed: *"Adding other attributes [predicates] based upon a certain column was really helpful, for example, I wanted to see the country or nationality of a person."* Also, the AND and OR operations of predicates in table columns were applauded by participants, especially in *"the initial population of columns"*-P1. However, participants had a difficult time to understand the "unionability score" that indicates the similarity between the found tables and the current table, especially for those without a computer science background, which were displayed next to the suggested tables. They were also confused about the missing entries in the table, displayed as "N/A" by the system. *"I saw some tables are missing data. I'm not sure if their Wikipedia pages themselves are not having the correct data, or some tables didn't have specific cell values."*-P12.

8.3.2 Guidance and Suggestion (G2). KTabulator's ability on providing guidance and suggestion made it easy for participants to create their desired tables. The predicate recommendation feature helped participants in two ways. First, it offered them a starting point when they were not certain. *"Sometimes it suggests things that you really don't initially have in mind. So I think, oh, this would be useful to include."*-P7. *"I think the most useful function is browsing through the attributes [predicates] in the very beginning. Because that's kind of the most frustrating thing you encounter when you're looking for data, like where to start."*-P9. Second, it helped participants find related information more effectively, such as *"We can easily select from the suggested attributes [predicates] to add more columns to build more comprehensive tables."*-P4. Also, *"Being able to actually find different attributes [predicates] for each of the individual [entities] was really useful, because sometimes I might not even know what I'm looking for, or what exactly I need for the question."*-P3. In addition to predicate recommendation, the table recommendation feature to union by KTabulator was appreciated. *"You have different* *tables and then you do it manually. [...] When you import one table, it recommends what other tables you can have."*-P12.

Together with the recommendation features, being able to get a preview of the information to add allowed participants to collect the right data efficiently. P11 commented: *"The right side shows you different attributes [predicates] you can choose from. So that's good because if you don't have any idea, you can just scroll down and see okay, I just select one and it shows a preview before you click on OK."* Also, P3 mentioned that *"When I saw the number between the brackets, I kind of realized that if I clicked on them, I wouldn't get that much information from them. So, I would avoid scrolling down to them."* On the other hand, P5, who rated 2 on Q13, never used this preview feature in any of the two tasks.

8.3.3 Information Browsing, Processing, and Sharing (G4). Overall, participants appreciated the available features in KTabulator for browsing, processing, and sharing the tables they created. They thought filtering and ranking table columns were essential. However, P1, P4, P5, and P9 required *"adding a range filter."* Some participants (P2, P3, and P8) also suggested to incorporate a feature for excluding columns and rows as well as manually editing and adding entries. About sharing, P3 provided a deep insight: *"In my field, being able to organize information and share it in a very intuitive way, is very important. So I would use it to look for tables that already exist in Wikipedia and try to aggregate them: comparing the gas taxes and perhaps finding a Wikipedia page that has all the different countries and all the different gas taxes."* Moreover, while most participants thought browsing Wikipedia page at the bottom panel was useful, P11 complained that *"I can not see much information for like three or five lines."* Similarly, P12 wanted the panel to pop up when double clicking on a cell. P3 (who rated 3 on Q12), in contrast, said that *"I personally didn't use the Wikipedia page preview a lot. I hid that most of the time, but I can see why it might be useful to have it."*

8.3.4 Comparison to Existing Tools. During the interview, we asked participants to compare KTabulator with any existing tool they would use for similar tasks, such as Excel or Google Spreadsheet. Most of them compared KTabulator with Excel and Google Spreadsheet, whereas some brought up certain data gathering APIs. While participants thought Excel and Google Spreadsheet are powerful and more familiar to them, they especially emphasized the data gathering features that are only available in KTabulator. *"If I need to find information from Wikipedia, a big advantage of this system is the speed. For Excel, I'd have to manually get the information that I'm looking for."*-P5. *"I like it takes off the workload. As I was saying, in Excel you have to do everything. This system does most of the work for you, and you can then use that to make a little bit of adjustments to fix it up."*-P7. Another advantage participants highlighted was the recommendation capabilities of KTabulator. *"Based upon a certain column, it recommends the next column, what you can add data according to the column. Whereas in Excel, you will have to search on your own and then copy into a cell."*-P12. However, they thought Excel has better formatting and data processing functions, such as filtering, sorting, pivoting tables, etc.

When compared with data gathering APIs, participants appreciated that KTabulator is easy to use, flexible, and effective to navigate information. For example, P2 said that *"There is probably an API for Wikipedia, but then there will be a lot of digging and filtering." "APIs are not as flexible, because you can get only a specific kind of information,"* also commented by P5. In addition, P10 mentioned: *"To get the information I would definitely use your tool over anyone else because it's pretty easy to navigate. I would get information, create a table from your tool, and then export it maybe to excel and run some other analyzes."*

9 DISCUSSION

While the results of the user study indicate the effectiveness of KTabulator for supporting participants to build ad hoc tables, the system still has some drawbacks. Our user study has revealed three broad usability challenges of KTabulator including table/data transformation, recommendation, and sparsity. We first discuss these challenges and then other limitations of KTabulator and our study.

Table Transformation Functions: Currently, KTabulator does not support advanced table operations, such as table pivoting, composing formulas, plotting charts, etc, which were requested or mentioned by several participants. For example, four participants requested a range filter on columns and two requested better sorting capability in the interview section. While our focus in this paper is about gathering data and building ad hoc tables, it is also clear that these spreadsheet functionalities are necessary to provide a better experience for users creating tables using KTabulator.

Data/Table Recommendation Searching: Three participants searched for tables to union with their tables but KTabulator failed to give suggestions. We later manually revisited these scenarios and found that if KTabulator had an adaptive way to expand its search for unionable tables, e.g., search for further degree neighbors of the Wikipedia table that users started from, or decrease its unionability score, the system could have found relevant tables to suggest.

Data Sparsity: Another usability challenge is handling of missing and wrong values, which are ubiquitous in data on the web. In several cases, participants saw many `N/A`'s after they populated a column. Although our data availability score in the preview feature warns them about the missing values, participants were still not satisfied when they saw many missing data values in cells. Moreover, three participants searched for data that did not exist in DBpedia (e.g., IMDb movie ratings). KTabulator naturally failed to provide any data. This part of the problem is inherent because KTabulator is designed to only work with Wikipedia and DBPedia and an important challenge is to provide access to other data sources. This will raise many further challenges, e.g., handling the data heterogeneity across sources.

Other Limitations: KTabulator is currently implemented with public endpoint of DBpedia, which has two drawbacks. First, it is not up to date with the most recent DBpedia version and contains some mismatches with some parts of Wikipedia. This did not create much problems in our user study. More important drawback is that the query execution times are rather slow, as P12 complained: *"If this could be made faster for long tables, that would (make your system more) helpful."* Instead of using the public DBpedia endpoint, we can easily extend our system to maintain a local RDF store to keep and update the knowledge graph and improve our query execution times. We plan to have our own knowledge graph store for this project in future. Additionally, we could improve the backend of the system to enhance the user experience by adding a query cache and progressively loading long tables.

Also, KTabulator does not support direct multi-hop exploration from a single entity. Users need to add one column at a time to do multi-hop exploration. We plan to add this feature in future. We can get design inspiration from the tool S-Paths [19], which allows direct multi-hop exploration by putting them in a drop-down list for selection. However, adding direct multi-hop exploration will add more complexity to KTabulator's user interface and needs further study to find a cleaner design.

Study Limitations: Our user study has few unavoidable limitations. First, we did not compare our system with any baseline systems because no prior system we are aware of provides the full functionality of KTabulator, though a baseline can be formed by allowing users to use a mix of systems, e.g., a SPARQL query system along with a spreadsheet software.

However, we did not think that is a fair comparison given that even these combinations could require a lot of copy pasting. Second, participants did not create tables based on their needs and on their own timeframes; instead, we designed T1 in a restrictive way and T2 with a predetermined topic. While this allowed us to better control what participants could do and thus derive consistent insights, we could miss many other factors about the usability of KTabulator. Third, our study was conducted in a lab setting and we did not have a large number of participants. Future development of KTabulator within a realistic setting would be necessary to investigate the usage of the system.

10 CONCLUSION AND FUTURE WORK

We addressed the problem of ad hoc table creation for personal use through a system that allows users to tabulate the structured information in DBpedia and Wikipedia. Many other tasks in practice require creating tables, such as data preparation for data science, which can be a very time consuming task. In this paper we focused on creating tables from the data in DBpedia-Wikipedia domain. However, our general design principles, such as the use of a dual knowledge graph-original data sources, or use of semantic information that is encoded in the ontology of the knowledge graphs, can be applied to extend KTabulator to create tables from other domains. In principle, it is possible to extend KTabulator to support using multiple knowledge graphs (and their original sources when available) from the *open linked data* space, which refers to the publicly available knowledge graphs and datasets. This is because, knowledge graphs in open linked data refer to each other using ontological predicates such as `owl:sameAs`, e.g., indicating that the entity `dbo:Christopher_Nolan` is the same as `wikidata:Christopher Nolan` in Wikidata or even the `http://d-nb.info/gnd/12394192X` entity in the German National Libraries catalogue that is exposed as a knowledge graph [34]. The Semantic Web [13], which lead to the emergence of linked data, is the vision to create a world-wide web of structured data that is accessible by software. We believe there is also tremendous value in putting humans in loop and facilitating humans to directly access the entire linked data through interactive systems. This would allow users to extract more complete tables and from a much wider range of domains. Doing so would also raise several important challenges, such as scalable ontology mapping, entity resolution, and querying of heterogenous knowledge graphs when same triples exist in multiple domains, which we identify as venues for future research.

ACKNOWLEDGMENTS

This research is supported by a grant from the Waterloo-Huawei Joint Innovation Laboratory.

REFERENCES

[1] 2004. Resource Description Framework (RDF): Concepts and Abstract Syntax. https://www.w3.org/TR/rdf-concepts/.
[2] 2008. About: The Dark Knight (Film). http://dbpedia.org/page/The_Dark_Knight_(film).
[3] 2013. OpenRefine. https://openrefine.org/.
[4] 2013. SPARQL Query Language for RDF. https://www.w3.org/TR/rdf-sparql-query/.
[5] 2020. Baidu Baike. https://baike.baidu.com/.
[6] 2020. What is a Knowledge Graph? https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/.
[7] 2020. Wikipedia: The Free Encyclopedia. https://www.wikipedia.org/.
[8] Faheem Abbas, Muhammad Kamran Malik, Muhammad Umair Rashid, and Rizwan Zafar. 2016. WikiQA—A question answering system on Wikipedia using freebase, DBpedia and Infobox. In *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*.
[9] Nafisa Anzum, Semih Salihoglu, and Daniel Vogel. 2019. GraphWrangler: An Interactive Graph View on Relational Data. In *Proceedings of the 2019 International Conference on Management of Data*.
[10] Maurizio Atzori, Giuseppe M Mazzeo, and Carlo Zaniolo. 2019. QA3: A natural language approach to question answering over RDF data cubes. *Semantic Web* 10, 3 (2019).
[11] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*.
[12] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. 2006. Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the 3rd International Semantic Web User Interaction Workshop*.
[13] Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The semantic web. *Scientific american* 284, 5 (2001).
[14] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*.
[15] Sören Brunk and Philipp Heim. 2011. tFacet: Hierarchical Faceted Exploration of Semantic Data Using Well-Known Interaction Concepts.. In *DCI@ INTERACT*.
[16] Diego Valerio Camarda, Silvia Mazzini, and Alessandro Antonuccio. 2012. LodLive, exploring the web of data. In *Proceedings of the 8th International Conference on Semantic Systems*.
[17] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2019. KBQA: learning question answering over QA corpora and knowledge bases. *arXiv preprint arXiv:1903.02419* (2019).
[18] Alessio De Santo and Adrian Holzer. 2020. Interacting with Linked Data: A Survey from the SIGCHI Perspective. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*.
[19] Marie Destandau, Caroline Appert, and Emmanuel Pietriga. 2020. S-Paths: Set-based visual exploration of linked data driven by semantic paths. *Semantic Web* Preprint (2020), 1–18.
[20] Bruno Dumas, Tim Broché, Lode Hoste, and Beat Signer. 2012. Vidax: An interactive semantic data visualisation and exploration tool. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*.
[21] Sébastien Ferré. 2017. Sparklis: an expressive query builder for SPARQL endpoints with guidance in natural language. *Semantic Web* 8, 3 (2017).
[22] Florian Haag and Thomas Ertl. 2014. Filter dials: combine filter criteria, see how much data is available. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*.
[23] Yeye He and Dong Xin. 2011. Seisa: set expansion by iterative similarity aggregation. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*.
[24] Philipp Heim, Sebastian Hellmann, Jens Lehmann, Steffen Lohmann, and Timo Stegemann. 2009. RelFinder: Revealing relationships in RDF knowledge bases. In *International Conference on Semantic and Digital Media Technologies*.
[25] Michiel Hildebrand, Jacco Van Ossenbruggen, and Lynda Hardman. 2006. /facet: A browser for heterogeneous semantic web repositories. In *International Semantic Web Conference*.

[26] Frederik Hogenboom, Viorel Milea, Flavius Frasincar, and Uzay Kaymak. 2010. RDF-GL: a SPARQL-based graphical query language for RDF. In *Emergent Web Intelligence: Advanced Information Retrieval*. 87–116.

[27] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

[28] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

[29] Xiao Ling, Alon Y Halevy, Fei Wu, and Cong Yu. 2013. Synthesizing union tables from the web. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[30] Steffen Lohmann, Vincent Link, Eduard Marbach, and Stefan Negru. 2014. WebVOWL: Web-based visualization of ontologies. In *International Conference on Knowledge Engineering and Knowledge Management*.

[31] András Micsik, Zoltán Tóth, and Sándor Turbucz. 2013. Lodmilla: Shared visualization of linked open data. In *International Conference on Theory and Practice of Digital Libraries*.

[32] John Morcos, Ziawasch Abedjan, Ihab Francis Ilyas, Mourad Ouzzani, Paolo Papotti, and Michael Stonebraker. 2015. Dataxformer: An interactive data transformation tool. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*.

[33] Fatemeh Nargesian, Erkang Zhu, Ken Q Pu, and Renée J Miller. 2018. Table union search on open data. *Proceedings of the VLDB Endowment* 11, 7 (2018).

[34] Deutsche Nationalbibliotek. 2016. The Linked Data Service of the German National Library. https://www.dnb.de/SharedDocs/Downloads/EN/Professionell/Metadatendienste/linkedDataModellierungTiteldaten.pdf?__blob=publicationFile&v=2.

[35] Heiko Paulheim and Lars Meyer. 2011. Ontology-based information visualization in integrated UIs. In *Proceedings of the 16th International Conference on Intelligent User Interfaces (IUI)*.

[36] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering table queries on the web using column keywords. *Proceedings of the VLDB Endowment* 5, 10 (2012).

[37] Vijayshankar Raman and J Hellerstein. 2001. Potters wheel: an interactive framework for data cleaning and transformation. *Working Draft* (2001).

[38] Juan F Sequeda and Daniel P Miranker. 2015. Ultrawrap Mapper: A Semi-Automatic Relational Database to RDF (RDB2RDF) Mapping Tool.. In *International Semantic Web Conference (Posters & Demos)*.

[39] Zhaohua Sheng, Xin Wang, Hong Shi, and Zhiyong Feng. 2012. Checking and handling inconsistency of DBpedia. In *International Conference on Web Information Systems and Mining*.

[40] Pavel Shvaiko and Jérôme Euzenat. 2005. A Survey of Schema-Based Matching Approaches. In *Journal on Data Semantics IV*. Vol. 3730.

[41] Paul R Smart, Alistair Russell, Dave Braines, Yannis Kalfoglou, Jie Bao, and Nigel R Shadbolt. 2008. A visual approach to semantic query design using a web-based graphical query designer. In *International Conference on Knowledge Engineering and Knowledge Management*. Springer.

[42] Ahmet Soylu, Martin Giese, Ernesto Jimenez-Ruiz, Guillermo Vega-Gorgojo, and Ian Horrocks. 2016. Experiencing OptiqueVQS: a multi-paradigm and ontology-based visual query system for end users. *Universal Access in the Information Society* 15, 1 (2016).

[43] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*.

[44] Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web (WWW)*. 771–782.

[45] Klaudia Thellmann, Michael Galkin, Fabrizio Orlandi, and Sören Auer. 2015. LinkDaViz–automatic binding of linked data to visualizations. In *International Semantic Web Conference*.

[46] Yannis Tzitzikas, Nikos Manolis, and Panagiotis Papadakos. 2017. Faceted exploration of RDF/S datasets: a survey. *Journal of Intelligent Information Systems* 48, 2 (2017).

[47] Hernán Vargas, Carlos Buil-Aranda, Aidan Hogan, and Claudia López. 2019. RDF Explorer: A Visual SPARQL Query Builder. In *International Semantic Web Conference*. 647–663.

[48] Denny Vrandečić. 2012. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*.

[49] Chi Wang, Kaushik Chakrabarti, Yeye He, Kris Ganjam, Zhimin Chen, and Philip A Bernstein. 2015. Concept expansion using web tables. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*.

[50] Shuo Zhang, Vugar Abdul Zada, and Krisztian Balog. 2018. SmartTable: A Spreadsheet Program with Intelligent Assistance. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

[51] Shuo Zhang and Krisztian Balog. 2017. Entitables: Smart assistance for entity-focused tables. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[52] Shuo Zhang and Krisztian Balog. 2018. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 World Wide Web Conference (WWW)*.

[53] Shuo Zhang and Krisztian Balog. 2018. On-the-fly table generation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

[54] Weiguo Zheng, Hong Cheng, Lei Zou, Jeffrey Xu Yu, and Kangfei Zhao. 2017. Natural language question/answering: Let users talk with the knowledge graph. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*.

[55] Erkang Zhu, Fatemeh Nargesian, Ken Q Pu, and Renée J Miller. 2016. LSH ensemble: internet-scale domain search. *Proceedings of the VLDB Endowment* 9, 12 (2016).

[56] Erkang Zhu, Ken Q Pu, Fatemeh Nargesian, and Renée J Miller. 2017. Interactive navigation of open data linkages. *Proceedings of the VLDB Endowment* 10, 12 (2017).

[57] Lei Zou, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over RDF: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*.