Vispedia*: Interactive Visual Exploration of Wikipedia Data via Search-Based Integration

Bryan Chan, Leslie Wu, Justin Talbot, Mike Cammarano, and Pat Hanrahan

Abstract— Wikipedia is an example of the collaborative, semi-structured data sets emerging on the Web. These data sets have large, non-uniform schema that require costly data integration into structured tables before visualization can begin. We present Vispedia, a Web-based visualization system that reduces the cost of this data integration. Users can browse Wikipedia, select an interesting data table, then use a search interface to discover, integrate, and visualize additional columns of data drawn from multiple Wikipedia articles. This interaction is supported by a fast path search algorithm over DBpedia, a semantic graph extracted from Wikipedia's hyperlink structure. Vispedia can also export the augmented data tables produced for use in traditional visualization systems. We believe that these techniques begin to address the "long tail" of visualization by allowing a wider audience to visualize a broader class of data. We evaluated this system in a first-use formative lab study. Study participants were able to quickly create effective visualizations for a diverse set of domains, performing data integration as needed.

Index Terms—Information visualization, Data integration, Wikipedia, Semantic web, Search interfaces

1 Introduction

There is great interest in making visualization available to a broader audience. Projects such as sense.us [15] and Many Eyes [31], enable collaborative analysis through easy-to-understand visualizations. Anyone can upload a data set, build a visualization, and then begin a discussion with others about their findings.

A major hurdle in using visualization in analysis is finding and preparing the data. Common operations are foraging for relevant data, integrating data from multiple sources, cleaning errors and inconsistencies, and converting to normalized formats. These transformations are expensive to perform and often require great judgment. They are particularly difficult if the input data is heterogeneous and does not conform to a well-defined schema. We believe that collaborative visualization software will be much more useful if it supports some of these early data integration tasks.

In this paper we present Vispedia, a system designed to support visualization of data interactively integrated from Wikipedia. We use Wikipedia because it contains a wealth of interesting semi-structured data for visualization, in the form of tables, infoboxes, lists, and hierarchies (categories). The data in Wikipedia is a hyperlinked graph, representative of the comprehensive, heterogeneous data sets being created by web communities.

We would like to support Wikipedia users who want to build ad-hoc visualizations to explore Wikipedia as a whole or to understand the context around a specific Wikipedia article; however, these users may be creating visualizations for unfamiliar data sets, where the structure of hyperlinks leading to additional, useful data is unclear. If the cost of integrating data and authoring a visualization is too high, they will abandon their attempts to create a visualization.

Vispedia aims to reduce this cost by including iterative data integration within the visual exploration loop, as shown in Figure 1. In previous work, we developed a keyword-query formalism and provided a non-interactive algorithm for ranking attributes against keywords [5]. In this paper, we describe an interactive system based on the formalism and evaluate its usability. Users begin building a visualization by browsing Wikipedia, selecting a seed table from an article, and choosing a visualization type. Then they use a keyword search to create a visualization and integrate additional data with the initial table. Vispedia provides a *query recommendation system* that suggests relevant data integration options using a novel, fast graph search over Wikipedia's semantic graph. In a first-use formative evaluation study of the system (n=7), participants successfully created compelling visualizations from Wikipedia data.

In addition to permitting users to rapidly create and refine standalone visualizations, Vispedia also supports new browsing and data integration scenarios. For example, users who create a visualization of data from a Wikipedia article can then follow links in the visualization back to related articles. Vispedia also makes the user-created augmented tables available for re-use, supporting future export to other visualization applications or integration back into Wikipedia.

After presenting a scenario to illustrate the main features of Vispedia, we describe the challenges presented by Wikipedia's complex schema and explain how we designed interaction techniques, algorithms, and the larger Vispedia system to address these challenges.

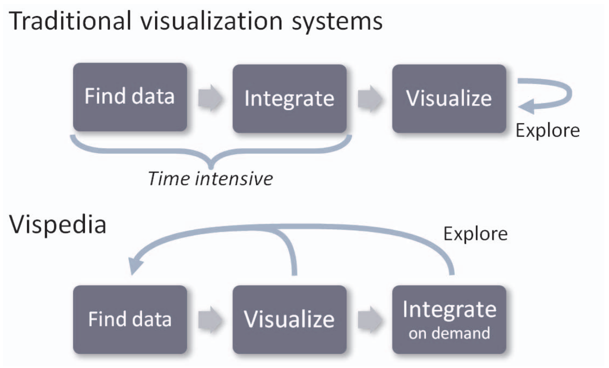
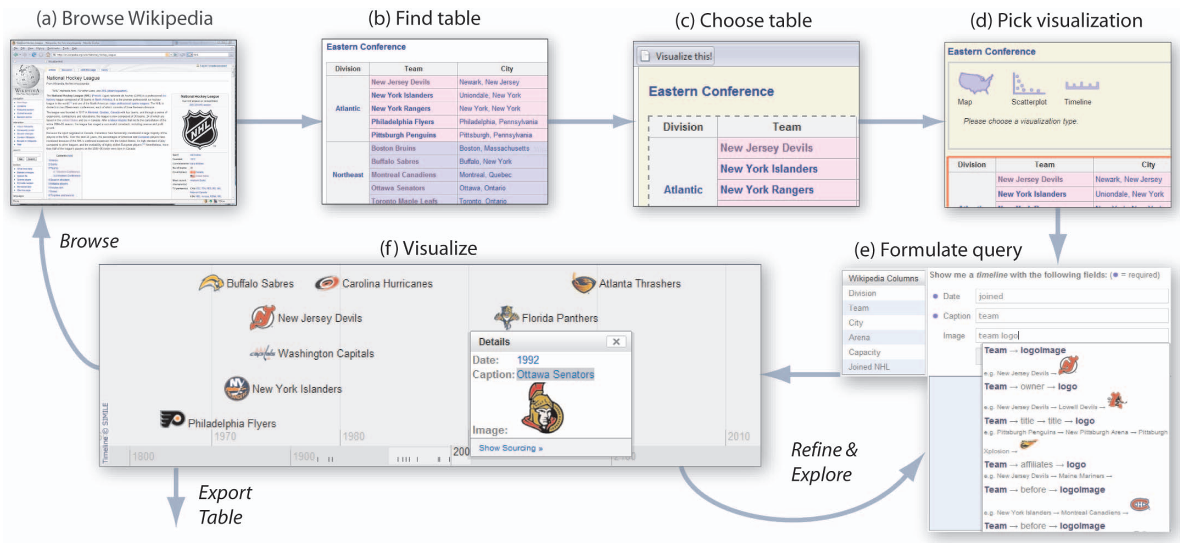


Fig. 1. (top) In existing systems, finding and integrating the data is a slow and necessary preprocess, so interactive exploratory visualization is limited to the data already integrated. (bottom) Instead, Vispedia uses a search interface to make visual exploration over large data sources with unfamiliar schema possible. Casual users can quickly find and visualize data, and then interactively integrate new data on-demand for a particular task.

*http://vispedia.stanford.edu

• *Bryan Chan, Leslie Wu, Justin Talbot, Mike Cammarano, and Pat Hanrahan are with Stanford University, E-mail: {bryanc,lwu2,jtalbot,mcammara}@stanford.edu, hanrahan@cs.stanford.edu.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.



Eastern Conference

Division	Team	City
Atlantic	New Jersey Devils	Newark, New Jersey
	New York Islanders	Uniondale, New York
	New York Rangers	New York, New York
	Philadelphia Flyers	Philadelphia, Pennsylvania
	Pittsburgh Penguins	Pittsburgh, Pennsylvania
Northeast	Boston Bruins	Boston, Massachusetts
	Buffalo Sabres	Buffalo, New York
	Montreal Canadiens	Montreal, Quebec
	Ottawa Senators	Ottawa, Ontario
	Toronto Maple Leafs	Toronto, Ontario

Wikipedia Columns: Division, Team, City, Arena, Capacity, Joined NHL

Show me a *timeline* with the following fields: (● = required)

● Date	joined
● Caption	team
Image	team logo

Team → logoImage
e.g. New Jersey Devils →
Team → owner → logo
e.g. New Jersey Devils → Lowell Devils →
Team → title → title → logo
e.g. Pittsburgh Penguins → New Pittsburgh Arena → Pittsburgh Xplosion →
Team → affiliates → logo
e.g. New Jersey Devils → Maine Mariners →
Team → before → logoImage
e.g. New York Islanders → Montreal Canadiens →
Team → before → logoImage

Fig. 2. Vispedia workflow: (a) While browsing Wikipedia, (b,c) a user finds a table and selects it using the Vispedia bookmarklet, (d) then picks a visualization type. On the Vispedia site, a list of table columns and the query recommender (e) help the user formulate an initial search query. Vispedia finds data matching the queries and creates a visualization (f). A user may then choose to browse back into Wikipedia, continue refining the existing query, explore related data using different visualization types, or export the augmented data table.

2 Scenario: NHL Teams

Carol is an avid fan of the National Hockey League (NHL). While browsing the main article on the NHL[1], she notices a table listing the teams playing in the Eastern Conference (follow along in Figure 2). Wanting to make a visualization of when teams joined the league, she clicks on the *Visualize this!* bookmarklet in her browser toolbar.

Doing so highlights the available data tables on the page and lets Carol pick the Eastern Conference table by clicking directly on it. Within the same Wikipedia page a menu appears allowing Carol to select an initial visualization type. This selection can be changed at any time, but seeing the original Wikipedia table in place may suggest starting points for exploration. For example, Carol notices the “Joined NHL” column, containing the date when teams first joined the league, and decides to create a timeline.

A new browser window opens showing the Vispedia site, with a query interface for specifying the visualization attributes needed by a timeline—*Date*, *Caption*, and *Image*. The data from the Wikipedia table is extracted by Vispedia and the available table columns appear in the box on the left of the screen. Carol can immediately click on a column like “Joined NHL”, to map that table column directly to a visual variable. When the original table does not contain a piece of desired data, such as the team logo, she can integrate additional data by typing in keywords to perform a search over the rest of Wikipedia. A query recommender (displayed in a drop-down under the query field) shows a ranked list of potential integration options with example data values. The list is dynamically updated as the user changes the query.

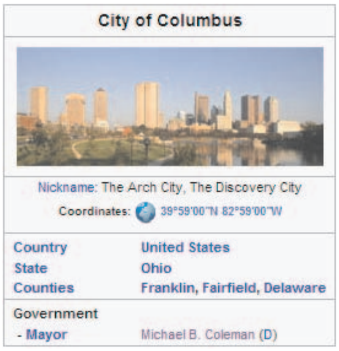
After creating a visualization, Carol may choose to refine it further. For example, she may modify the query in order to change the images used by the visualization or she might change the visualization type. Alternatively, Carol can click on items in the visualization or fields in the table rows (not shown) to see more details, and browse back into Wikipedia to learn more.

In a few minutes, Carol created a useful visualization from Wikipedia. During this process, she interactively performed complex data integration tasks using a simple, easy to understand interface.

3 Data Integration Challenges

Data integration is the process of mapping several data sources into a common schema [21]. In Vispedia this happens by combining information from multiple Wikipedia pages into a single table. Data integration is a difficult problem. Sources can have different schema with subtle variations in the meaning of attribute labels. In addition, the quality of data varies from source to source. Some sources are more complete, more relevant, or more trustworthy. Data integration involves extracting structured data from sources, building queries mapping each source to the common schema, and choosing between different sources based on criteria like data quality.

[1]http://en.wikipedia.org/wiki/NHL


City of Columbus

Nickname: The Arch City, The Discovery City
Coordinates: 39°59′00″N 82°59′00″W

Country	United States
State	Ohio
Counties	Franklin, Fairfield, Delaware
Government	
- Mayor	Michael B. Coleman (D)

Fig. 3. Part of the Wikipedia infobox for the city of Columbus, Ohio. Infoboxes contain some of the structured information available on Wikipedia.

Vispedia addresses the challenge of building queries and choosing between alternative queries based on relevance. Other measures like completeness and trustworthiness are left for future work. We also depend on external sources for most of the data extraction. While the system extracts seed tables, we use the semantic graph from DBpedia that includes information from Wikipedia infoboxes [4].

In this section we will introduce the infobox, explain how the search for attributes corresponds to path queries on the infobox graph, and show why integrating data from this graph is hard.

3.1 Infobox Graph

Infoboxes contain a list of labeled attributes and appear in about 600,000 Wikipedia articles. The infobox for Columbus, OH (Figure 3) links to other Wikipedia articles, as well as images and literal values like population. The infobox data forms a semantic graph where objects (articles, images, and literals) are nodes and relationships between objects are directed edges. For example, in the Columbus, Ohio infobox, one of the extracted edges links the city to its mayor:

$$\text{Columbus, Ohio} \xrightarrow{\text{leader name}} \text{Michael B. Coleman}$$

Each infobox is based on a template that lists possible attribute labels. For example, the edge label "leader name" comes from the infobox template for settlements[2] and appears in the wiki markup for Columbus, Ohio infobox. Wikipedia translates this label to the more readable ”Mayor” for page display.

3.2 Path Queries

Following a path in the graph is like following hyperlinks between Wikipedia articles. *Paths* correspond to indirect relationships between nodes in the graph, like the political party of Columbus' mayor:

$$\text{Columbus, OH} \xrightarrow{\text{leader name}} \text{Michael B. Coleman} \xrightarrow{\text{party}} \text{Democratic}$$

While keywords describing an attribute could be turned directly into corresponding paths, we use *path queries* as an intermediate step to provide a higher-level summary for the user and to improve path search efficiency.

A path query describes an abstract relationship, for example, between a city and the political party of its mayor:

$$? \xrightarrow{\text{leader name}} ? \xrightarrow{\text{party}} ?$$

We formerly defined the terms *instance path* and *schema path* [5], but we will now use the more common terms path and path query respectively.

For example, say the user wants to run the above path query against a table of cities, and Columbus, OH is one city in the table. Then the system will try to match the following pattern in the graph and return the path shown at the beginning of this section.

$$\text{Columbus, OH} \xrightarrow{\text{leader name}} ? \xrightarrow{\text{party}} ?$$

In summary, Vispedia uses an improved algorithm to solve the same problem that we introduced in previous work: turning a keyword search into a list of possible path queries ranked by relevance, then using this list to match for paths in the graph, producing data for the visualization. When several paths match a query, our system currently presents only the first match. We do not yet handle 1-to-many relationships.

3.3 Complexity of Available Path Queries

The high branching factor and heterogeneous structure of the graph gives a large number of possible path queries. This makes it difficult for a user to discover good queries for a visualization attribute. It also makes the problem of finding and ranking path queries difficult for the system.

3.3.1 High Branching Factor

The articles in Wikipedia are highly interconnected [20]. In a random sample of 1% of the articles with infoboxes, each article has 19 different attributes on average. This means that for a single article, the number of possible path queries grows exponentially with path length.

3.3.2 Diverse Infobox Templates

More importantly, although infobox templates introduce some structure to the schema, there is a great deal of variety between articles. Templates define what relationships an infobox might contain; however, Wikipedia is authored by many contributors who have only loosely standardized on templates for each domain – There are 2139 templates in our data set using over 8000 different edge labels. Articles on two people can use different templates. For instance, the article on John McCain uses a senator template while Kevin Bacon has an actor template. Labels used for similar relationships can vary between templates, so while the Senator template uses $\xrightarrow{\text{birthPlace}}$, the actor template uses $\xrightarrow{\text{placeOfBirth}}$. In some articles, template fields are not completely populated, leading to missing data. Furthermore, unlike strict ontologies, templates specify attribute labels, but don't constrain the type on attribute values.

All of these factors – many branching edges, semantically similar labels, loose template constraints, and missing data – contribute to the variety of path queries available. Let us return to the NHL example and enumerate the number of path queries available across the set of fifteen teams:

Path Length	Average per Team	Total Distinct for All Teams
1	37	84
2	910	3,907
3	15,366	111,525
4	404,497	3,839,530

The second column shows the average number of available path queries for a single team. This exhibits the exponential branching previously mentioned. In comparison, the third column shows the total number of different path queries across all teams. This is the space of queries the user must consider when finding data to visualize. Due to differences between teams, the total number of queries is much greater than the average for any single team. Some of these queries match paths for many teams, but a large number match paths for only a few teams.

3.4 Challenges

The space of possible path queries is large, unfamiliar, and different from one object to the next. Because of the semantic subtleties, any approach for matching path queries to keywords cannot be fully automatic. We need to help the user understand and choose path queries from this large and complex space. This poses both user interface and system challenges.

4 The Design of Vispedia

We have now described the challenges involved in integrating Wikipedia data. In this section we first describe the design goals and criteria that we developed while working on Vispedia. We discuss how these criteria are derived from the principal challenges that arise from using an integration-on-demand approach to visualization. We then describe the design decisions we have made to meet these criteria.

4.1 Users

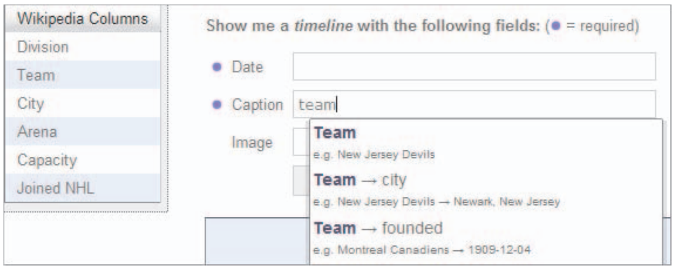
We would like our tool to be used by Wikipedia readers who would benefit from ad-hoc visualizations while browsing, often in domains outsider their area of expertise. We would also like our tool to be used by Wikipedia editors, who may want to quickly create visualizations to accompany Wikipedia articles.

Both classes of users are fairly nontechnical, so the interface should be straightforward. We assume no knowledge of database technology or terminology, such as "schema integration," table joins, or SQL. It should be possible to create a visualization in a small number of steps, both as a starting point for iterative refinement, and to support the common case where the first version is good enough. More abstract data integration tools should only appear if user needs to refine the visualization further.

4.2 Designing a Search-Based Interface

As discussed previously, the space of paths within Wikipedia is very large. To permit users to effectively find useful attributes using these paths, we propose a design that narrows the search space to be anchored to a specific, user-specified seed table. A simple query interface then hides the complexity of ranking different paths.

[2]http://en.wikipedia.org/wiki/Template:Infobox_Settlement


Wikipedia Columns
Division
Team
City
Arena
Capacity
Joined NHL

Show me a *timeline* with the following fields: (● = required)

● Date

● Caption team

Image

Team
e.g. New Jersey Devils

Team → city
e.g. New Jersey Devils → Newark, New Jersey

Team → founded
e.g. Montreal Canadiens → 1909-12-04

Fig. 4. Vispedia search interface showing a list of columns pulled from the user-selected Wikipedia column and the query recommender showing data items that are available through search.

In this paper, we focus on helping the user understand the space of path queries and handle the differences moving from one table to another. For differences within a table, where one row requires a different path query than another, we continue to use our previous method: automatically falling back on lower ranked alternatives [5].

We will consider our design within the user-interface framework proposed by Shneiderman, where search starts by formulating and creating a query, then reviewing the results, and refining the query [26].

4.2.1 Picking a Topic

In Shneiderman's framework, query formulation is the most complex phase, involving the need to decide where to search, what terms to search for, and what variants of those terms to accept. Vispedia's starts the user's search at a table selected with the *Visualize this!* bookmarklet.

After selecting a table, the user is presented with a box above the table showing a selection of visualization types supported by Vispedia: map, timeline, and scatterplot. Seeing the table in-situ within the article and choosing a visualization type while looking at the article allows a user to build an intuitive understanding of the available data. Picking a table and visualization type first also narrows the search space since Vispedia can start from relatively complete seed data and it can search only for the data types required by the visualization type.

4.2.2 Formulating a Search

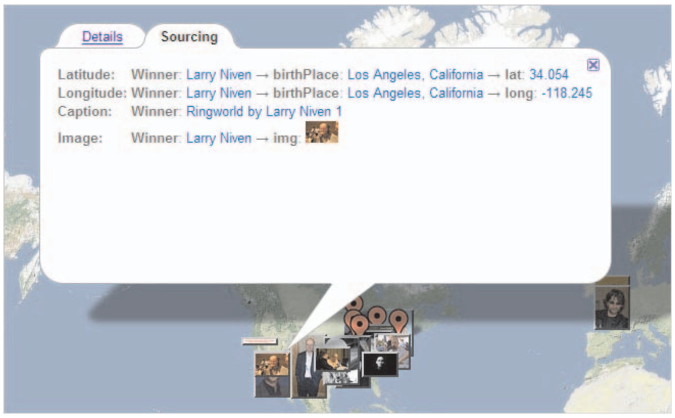
Once a user picks a table and a visualization, they need to query Wikipedia for the attributes needed to create the visualization. The space of possible path queries is large, so to simplify query formulation, the interface helps the user discover possible queries.

In many cases, the values needed are in the table itself. To use these values for a visual attribute, the user simply clicks on the table column in a list near the query fields (see Figure 4). This populates the query with the column name and causes the search to use data from the table, unless data is missing or does not have the right data type for the visualization.

If the user does not find the data they need in a table column, users can type in a list of keywords. Vispedia will then try to match the keywords to relevant path queries.

We assume that the user is not familiar with the structure of the Wikipedia hyperlink graph and the information available from following paths in the graph. In order to find the correct attributes, we provide a query recommendation engine similar to an auto-complete widget (also shown in Figure 4). This interactive feedback makes it possible to quickly see if the search will match useful results and helps suggest additional terms to formulate or refine the query.

The widget shows a ranked list of paths, along with an example data instance that results from following each path. For example, the query recommender, shown in Figure 4, displays three possible paths related to the query "team" in sorted order, where the first path is estimated to be the most relevant. The second path in this list,"Team → city", shows the matching example for one NHL team *"New Jersey Devils → Newark, New Jersey"*. The user can select an item directly from the query recommender. This changes the keywords to the terms from the path query, guaranteeing that the chosen path query will then be used first as the most relevant match.


Details | Sourcing

Latitude: Winner: Larry Niven → birthPlace: Los Angeles, California → lat: 34.054
Longitude: Winner: Larry Niven → birthPlace: Los Angeles, California → long: -118.245
Caption: Winner: Ringworld by Larry Niven 1
Image: Winner: Larry Niven → img:

Fig. 5. Users can click items in the visualization to reveal additional sourcing metadata. This permits users to verify and evaluate how Vispedia is locating data in Wikipedia.

4.2.3 Reviewing Search Results and Refining Query

The next phase in Shneiderman's framework is reviewing the results. After the user fills in desired fields of the visualization template, they can click "Show Visualization" which causes Vispedia to query the Wikipedia semantic graph. Returned data are shown both on the visualization and in tabular form. Showing both representations makes incorrect or missing results easier to see.

The paths used to find the data are available by clicking on columns in the table or by clicking on items in the visualization (see Figure 5). These paths provide links back to Wikipedia which users can use to better understand how Vispedia is finding the results. This allows the user to evaluate the provenance of the data and then revise the query to improve the results.

5 Implementation Details

Here are the implementation details for the user interface, data set, and table extraction.

5.1 Frontend

The bookmarklet used for table selection is implemented in Javascript. The web frontend is implemented in PHP, and the visualization interface uses Javascript components from MIT's Simile timeline [24], Google Maps [13],

5.2 Graph Data

The frontend accesses the graph data using REST HTTP requests to a mod_python Apache module.

For the data set, we use version 3.0 of DBpedia [4], extracted from Wikipedia in January 2008. This graph has 13 million nodes and 36 million edges taking up 4GB of storage. It includes the infoboxes, redirections, images, and geocoordinates data from DBpedia. The redirections handle alternate names for the same object, and images and geocoordinates hold pictures and locations if they exist for an article.

While there are database solutions for storing and querying semantic graphs [18, 2], they are not designed for the sort of best-first path search operations we depend on. Instead, we keep all the graph edges in a memory mapped adjacency list, with a Berkeley DB for holding strings from node and edge labels.

5.3 Linking Table and Graph

Unfortunately DBpedia depends on Wikipedia dumps, which are often several months out of date. Vispedia combines the current version of a Wikipedia table with the DBpedia semantic graph. This implementation allows a user to start from the live, immediately editable data they see on Wikipedia, and serves as a model for future combinations of arbitrary web tables with Wikipedia data.

The system downloads the user-selected table from Wikipedia, parses the wiki markup, and converts it to a graph. This graph contains a node for each table row, with labeled edges to each field in the row. For example, in the NHL scenario table, Figure 2, the following edge is extracted from the Team column of the first row:

$$\text{row:0} \xrightarrow{\text{Team}} \text{New Jersey Devils}$$

When the DBpedia graph and a table graph reference the same Wikipedia article or use the same string literal value, then the graphs share a node and the search can follow paths from the table into Wikipedia to find additional data.

6 Path Search Algorithm

Vispedia uses a fast path search to recommend path queries and match data for visualization. As in our previous work [5], the path search turns a user's keywords into a ranked list of the most relevant path queries; however, the current graph search provides answers of similar quality significantly faster than our old non-interactive algorithm. The main improvement is a reformulation of the ranking function to allow a time-limited A* search. This is a best-first search that explores available path queries in order of decreasing relevance. It is both more efficient and more likely to return good answers within a given time limit.

6.1 Ranking Function

The path search shows the user a list of path queries ranked by relevance. We focus on supporting interactive response times with a fast, simple algorithm that gives usable, but not perfect quality. There are ways to improve match quality, such as corpus based techniques [22], but these are slower and more costly to implement.

The function for ranking path queries against keywords is the same as in our previous work, except that we have simplified the heuristic for penalizing longer path queries. We keep the same cap on the branching factor and continue to ignore links through non-string literals. At the moment, the interface does not allow the user to manage path queries with multiple matches, so we no longer use the common path or voting heuristics.

The updated ranking function is motivated by document similarity metrics based on term frequency vectors [25]. The user's keyword search and the path query are each broken into a bag of words by considering non-alphanumeric separators and camelCasing. Words are case invariant, but may appear multiple times in a bag. For example, the path query $? \xleftarrow{\text{Team}} ? \xrightarrow{\text{teamLogo}} ?$ becomes the bag of words $\{\text{team}, \text{team}, \text{logo}\}$

For each bag, we compute a word count histogram, called a term frequency vector:

$$\text{tfv}_{\text{bag}} = \{t_1, t_2, \cdots, t_n\}$$

where n is the number of different words that appear in graph edge labels, and t_i is a count of how many times the word with index i appears in the bag. The term frequency vector for the path query above will have a count of 2 for the word "team", a count of 1 for "logo", and all other t_i will be zero.

A Manhattan distance function between term frequency vectors [25] gives the similarity between the user's keyword search, $\text{tfv}_{\text{keyword}} = \{q_1, q_2, \cdots, q_n\}$, and the path query, $\text{tfv}_{\text{path}} = \{p_1, p_2, \cdots, p_n\}$:

$$\text{dist}(\text{tfv}_{\text{keyword}}, \text{tfv}_{\text{path}}) = \sum_{i=1}^{n} \text{tfidf}(i)\,\text{w}(p_i, q_i)|p_i - q_i|$$

$$\text{where } \text{w}(p_i, q_i) = \begin{cases} \alpha & \text{if } p_i > q_i \\ 1 & \text{otherwise} \end{cases}$$

As in previous work, we use *tf-idf* weights to give common terms like "of" and "the" less impact on the distance. The weighting function, w, penalizes words in the path query that do not appear as a keyword, thus favoring shorter paths. Currently α is set to 0.5, but we have not tested this extensively.

6.2 A* search

In our previous paper, we used an exhaustive graph search that took several minutes to find paths to a depth of 4 for a set of 101 U.S. senators [5]. The revised distance function presented above permits Vispedia to use an A* search that explores the graph in approximate order of relevance, greatly improving performance.

We run a separate A* search starting from each seed table row and maintain the 5 best path queries per row. Results from all rows are collected together for display in the query recommender. The ranking function does not always match human judgment, so keeping the top 5 path queries provides a reasonable but not overwhelming number of additional options for the query recommender.

During the search, the algorithm estimates the relevance in unexplored parts of the graph. It explores the parts with the highest relevance first and prunes parts where there is no chance of finding better path queries. In order to prune correctly, we need a bound on how much closer a path can get to the keywords by adding more edges. Adding edges that match more keywords can decrease the distance, making the path more relevant. It is easy to see that if

$$\text{h}(\text{tfv}_{\text{keyword}}, \text{tfv}_{\text{path}}) = \sum_{i=1}^{n} \begin{cases} \text{tfidf}(i)\alpha(p_i - q_i) & \text{if } p_i > q_i \\ 0 & \text{otherwise} \end{cases}$$

then

$$\text{dist}(\text{tfv}_{\text{keyword}}, \text{tfv}_{\text{path}^*}) \geq \text{h}(\text{tfv}_{\text{keyword}}, \text{tfv}_{\text{path}})$$

for all paths, $path^*$, which include $path$ as a prefix. Thus h is exactly the bound we need, and it is only achieved if the extended path matches every keyword.

6.3 Time Limited Search

Pruning in the A* search usually improves performance, however if the search discovers no good path queries, it cannot prune effectively. This can happen if the user enters a keyword that does not appear in the graph.

To ensure the search still returns the best possible answer interactively, we run A* searches one row at a time and limit the search time to 0.1 seconds per row and 1 second overall. This means that the space of paths for a single row may not be fully explored, missing parts with less estimated relevance. Also some rows may not be explored at all if the search reaches the overall time limit. This can be a problem if an row omitted during the search needs a unique path query to find data. Usually though, the path queries from other rows can be reused to match data for the omitted row.

6.4 Performance

We will evaluate the proposed distance metric and A* search based on the number of path queries considered by the search and the execution time compared to an exhaustive search. Evaluation of the result quality was done as part of the user study.

6.4.1 Search Space

The table below shows the average number of path queries considered at each length over all rows of the NHL table, with each column showing a different query. Note that the first step of any path query always passes through one of the 7 columns in the Wikipedia table.

Length	"Joined NHL" (date)	"Team" (string)	"Team logo" (image)
1	7	7	7
2	200	186	204
3	2314	268	1407
4	3274	255	1499
5	136	2	390

The search for "Joined NHL" matches the table column, but then finds few other paths matching the rare keywords. As a result, the search to find the top 5 paths continues with few opportunities to prune early. In general, pruning during the A* search avoids the exponential growth in the number of longer path queries that appears in an exhaustive search.

6.4.2 Search Time

The next table shows the total execution time of the A* search over all rows of the NHL table. Since the table is relatively small, all the items are fully explored well under the time limit of one second. For comparison, we also show the speedups compared to an exhaustive depth first search of all path queries up to length 4, which takes 3.5 seconds.

	Time(s)	Speedup versus DFS to length 4
"Joined NHL" (date)	0.34	10
"Team" (string)	0.026	130
"Team logo" (image)	0.13	26

6.4.3 Effect of Time Limits

Over two weeks that included the user study as well as our own usage, we logged a larger sample of 526 unique queries on 37 different tables from our website. Most of the queries finished within the time limit (457 queries). The remaining 157 queries that reached the 1 second barrier tended to be from larger tables, with over a hundred rows, and these searches still managed to fully explore 38% of their rows on average.

Partial exploration will miss candidate path queries that exist only for a small number of rows, but we have observed that these tend to be less useful than more frequent candidates.

7 Usability Evaluation

We conducted a first-use study to qualitatively evaluate usability. There were 7 total participants: 2 female, 5 male. Their ages ranged from 18-24 (five participants) to 25-34 (two). Most participants (over two-thirds) used Wikipedia on a daily basis, and had no experience with database technologies such as SQL or Microsoft Access.

The first two participants took part in a pilot study, designed to uncover major usability flaws and provide incremental feedback on Vispedia's design.

7.1 Study Protocol

Each study session took approximately 60 minutes. Participants were seated at a workstation and used a standard web browser that had the "Visualize this" bookmarklet pre-installed. Direct observation and server query logs were used for analysis.

We demonstrated Vispedia's interface, briefly walked through the process of specifying and refining search terms, and asked participants to complete three design tasks of increasing complexity. The first task tested the basic usability of our system, and instructed participants to create a timeline of NHL hockey teams and their logos, according to when teams joined the NHL. In the the second task, participants were asked to plot all fourteener mountain peaks in California on a map and use this depiction to plan a climbing trip beginning with the southernmost peak. Both of these tasks required data beyond the initial seed table, and the second task involved more complex queries.

Finally, the third design task was open-ended, asking users to pick a topic of personal interest and to create a compelling visualization from a table found on Wikipedia. We suggested potential topics, but participants generally diverged from this list, preferring to explore, for example, topics ranging from the Roman Empire to aeronautic disasters of the twentieth century. The table sizes varied from five rows (Roman Emperors from the Julio-Claudian dynasty) to 577 rows (United States Micropolitan Statistical Areas).

	Time (min)		Display count	
	mean	σ	mean	σ
Task 1 (NHL)	5.2	2.7	3.7	2.3
Task 2 (14ers)	3.7	2.3	2.3	0.8
Open Task	2.4	3.3	1.8	1.6

Table 1. Time spent and number of times a revised query was displayed per visualization, taken from server logs. These summaries omit one participant on Task 1 and one on Task 2, who took more than 20 minutes due to difficulties refining the query to answer the given questions.

7.2 Results

Overall, results from evaluation study support our hypothesis that the Vispedia system and associated interaction techniques were easily learnable (mean=4.1, median=4 on a 5-point Likert scale, σ=0.7). In combination with a novel, fast path search algorithm, the resulting system was found to be usable by first-time users (mean=4.3, σ=0.8), who were able to browse or search for tabular data sets and quickly prototype visualizations.

In the following analysis, a visualization begins with choosing a seed table and visualization type and includes all subsequent revisions to the keyword searches.

In the first two tasks, most participants worked on a single visualization. One participant attempted several different seed tables on the California fourteeners task, as some were not sufficient to complete the task given. While all participants completed the first task, not all participants found the southernmost fourteener in the second task.

In the open task, we encouraged participants to build additional visualizations if time permitted. Participants attempted 19 visualizations using 15 different seed tables. 4 of these failed to yield useful visualizations due to missing data, either because the seed table lacked enough hyperlinks, the linked articles contained no infoboxes, or our table extractor parsed the wiki markup incorrectly.

7.2.1 The cost structure of visual sensemaking

Participants were able to quickly create visualizations. The average time spent per visualization in each task was generally 5 minutes or less, as seen in subsection 7.2.

One participant wrote that Vispedia could help in "sparing the user from the tedium of manual spidering" when one is interested in information not presented in the original Wikipedia table.

Participants generally agreed that the Vispedia system responded quickly enough (mean=4.1, σ=1.1). Furthermore, they strongly agreed that Vispedia "facilitates rapidly exploring different visualizations and data" (mean=4.4, σ=0.5).

7.2.2 Data integration on demand

Before running the lab study, we hypothesized that Vispedia would support data integration and visualization on demand, by allowing a fast, iterative, and visual sensemaking loop. Observing our participants as they completed the provided design tasks, we noted that participants would enter a few keywords for a subset of fields, display the corresponding visualization, and then return to fill additional fields or refine the keyword search, much as one would refine a Web search. Participants updated keywords and displayed visualizations multiple times per task as shown in subsection 7.2.

During the open task, in 10 out of 15 successful visualizations, users integrated data via the path search. In the remaining 5 cases, the original table contained all the desired data, and these visualizations were completed very quickly (mean=0.4min, σ=0.2min). This shows that

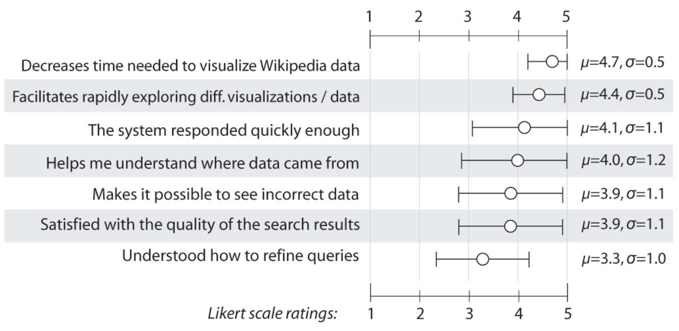


Fig. 6. Post-experiment questionnaire results. Error bars indicate one standard deviation in each direction.

while Vispedia enables integration on demand, it also efficiently supports traditional visualization from structured tables.

Participants generally agreed that they were satisfied with the quality of the search results returned (mean=3.9, σ=1.1), and were able to perform data integration and visualization for tables the authors had not previously encountered. One participant said they thought it was “impressive that the system did the smart thing in most cases.”

7.2.3 Data quality and provenance

While integrating data from multiple sources, participants often used the visual representations of data to heuristically check data quality. For example, when plotting data specific to one country or state, participants easily noticed visual outliers. In contrast, participants performing the California fourteeners task often failed to recognize numeric outliers if they inspected data mainly in the table form.

Participants mostly agreed that the Vispedia “makes it possible to see incorrect data” (mean=3.9, σ=1.1) and also “helps [them] understand where data comes from” (mean=4, σ=1.2). We observed that participants made frequent use of the ability to quickly toggle between “data sourcing” view and data view in the table provided, as a way to quickly get a sense of where and how different items are being sourced.

7.2.4 Compelling visualizations

Participants were able to create compelling visualizations of their own choosing, and several participants wanted to e-mail visualizations they created to friends. One participant thought Vispedia was “an easy way to share graphs... can argue points in blogs visually.”

7.3 Shortcomings

7.3.1 Finding seed data

In the open task, participants employed various strategies for finding data. They encountered a number of difficulties and spent between 3-10 minutes (mean = 7.1min) finding a table. Some common problems included the inability to search for tables, difficulty browsing to an article with a table, and incorrect extraction of complex table formats.

Possible solutions to this problem include building search engines for tabular web data and providing additional information scent to show data sets related to an article. In addition, with an entity reconciliation step, any table could be linked to Wikipedia, increasing the number of potential starting points.

7.3.2 Learning the query model

While the majority of participants reported that they “understood how to refine queries,”, other participants either disagreed or were neutral about this claim. All of the participants were able to refine queries and author visualizations, so we infer that participants may have struggled to align their mental model of Wikipedia’s semantic graph (infoboxes, tables, and entries) with system’s understanding of Wikipedia. For example, a NHL logo image may be labeled “logo_image” in the infobox template, but neither “logo” nor “image” may appear in the rendered Wikipedia article itself.

To address this mismatch, a future system may provide a way to co-locate semantic metadata with the original web page. In particular, the bookmarklet could also allow users to mouse over wiki elements which have invisible semantic representations, surfacing them visibly.

7.3.3 Transforming and Tailoring

Similarly, participants were split as to whether Vispedia allowed them to “specify data [they] want to show” on the visualization (mean=3.3, σ=0.8). At times, participants wanted to plot “unrelated” images or iconic representations such as triangles or arbitrary pictures from Wikipedia. One participant asked for the ability to edit the table sourced from Wikipedia directly. These suggestions would improve users’ ability to transform and re-represent sourced data. They also suggest the tailoring and editing of existing visualizations as one avenue for future work.

7.3.4 Change blindness

Because data integration happened on demand, participants would occasionally exhibit change blindness when refining queries. That is, they were unable to tell whether or not changing keywords or path queries actually resulted in a different integrated data set. Participants also failed to notice missing data when only one or two cells were incomplete. More work needs to be done in understanding how visualization systems can help highlight missing data and combat change blindness.

7.3.5 Sharing and Collaboration

Multiple participants requested the ability to share and annotate their visualizations. Techniques described in Heer’s sense.us [15] and ManyEyes could be extended to a system that closes the loop by allowing users to embed created visualizations or tables back into the original data source (in this case Wikipedia).

8 Related Work

Vispedia builds on the research in two areas: web-scale collaborative data exploration, and semantic graphs.

8.1 Collaborative Data Exploration

Collaborative visualization projects such as sense.us [15] and Many Eyes [31], as well as commercial endeavors like Swivel [29] and Tableau Server [30] have made it easy to explore and share well structured data with other users. GraphWise [14] is another example that combines user-created content with visualizations automatically generated from Wikipedia tables. These systems depend on structured data with a known schema, so Vispedia complements these advances in social visualization by addressing the case where the schema is large and unknown.

Visualizations that make transparent the social dynamics of Wikipedia authors, like WikiDashboard and the revert graph [1, 28] help users better understand the source of the data itself.

The challenge of performing data integration at web-scale has been explored by others. The Cimple [6] project created a web platform for supporting the social aspects of community-driven data integration. Madhavan et al. promote a formal “Pay as You Go” [23] model where web-scale data integration is done on demand and implicit or explicit user feedback is incorporated to improve the results. Like our project, they use queries and ranking to enable integration. Google Base [12] and FreeBase [9] are two commercial ventures whose goal is to create large collaboratively-authored semantic databases.

In contrast to these systems, Vispedia emphasizes visualization as the major focus, and as part of a sensemaking loop rather than as the end point of a data integration pipeline.

8.2 Semantic Graphs

A second rich research area is the work on how to extract, search, and visualize semantic graph data.

There has been growing interest in extracting semantic relationships from Wikipedia. DBpedia provides an extractor for the infoboxes [4] and links the graph they extract with a wider Linking Open Data community project [3]. The KYLIN system uses the relationships in infoboxes to seed a relationship extractor on the free text of Wikipedia [34]. The YAGO system seeks to build a better category hierarchy, a first step to more meaningful classes of objects on Wikipedia, by combining the Wikipedia category hierarchy with the Wordnet hierarchy [27]. Dontcheva et al. [8] have suggested a personal data extraction and integration system based on templatized screen scraping. Making more organized, interlinked semantic data available increases the benefits of building visualizations from such data.

Many query interfaces and search algorithms exist for semantic graphs. There are graphical query interfaces for RDF such as the W3C's Tabulator [33], and the basic DBpedia search interface [4], use a link following model similar to web browsing. The W3C SPARQL query language [32] provides a way to specify graph patterns against a semantic graph.

Compared to the stepwise query interfaces found in all of these systems, we support a more flexible keyword search for discovering and matching paths, more akin to the proximity search system in Lore [10], which ranks objects based on their distance in the graph to focus objects, or The DBpedia Relationship Finder [20], which lets users browse paths connecting pairs of nodes.

The graph search algorithm we present is similar in spirit to the Bidirectional keyword search by Kacholia et. al. [19]. They developed a method for matching keywords to subtrees that searches the graph as directed by a scoring function. Indexing schemes for keyword search on heterogeneous graphs [7] and for summarizing relationships in DataGuides [11] also exist; however, it is these approaches may not scale for path queries on a schema as complex as Wikipedia's.

Exhibit is a data publishing system that includes an HTML-tag based method of specifying a visualization from an underlying graph [16]. Potluck allows data integration between several Exhibit data sources [17]. Vispedia aims instead give a wider audience access to data integration on web-scale data sources.

9 Conclusions and Future Work

We introduced a novel system, Vispedia, for building visualizations sourced from Wikipedia, and described its algorithmic and interaction design. Vispedia addresses the long tail of visualization, by reducing the work required to integrate and visualize data.

In a first-use lab study with 7 participants, we found that participants did indeed perform data integration and visualization on demand, switching back and forth between visual and tabular representations. The study also demonstrated the usability of Vispedia and the interaction techniques it embodies, as well as the feasibility of an interactive visualization authoring tool, enabled by a fast, novel graph-structured search algorithm.

We hope that future visualization researchers consider the challenges involved with data sets such as Wikipedia, and continue to work toward a tighter collaborative sensemaking loop, which seamlessly blends foraging for data, data integration, visual and non-visual sensemaking, re-representation, and storytelling in a compelling fashion. Exposing more data without a pre-specified schema changes the fundamental assumptions of visualization and suggests many other open problems: Extraction and integration interfaces like Vispedia can be made more powerful yet more usable; missing data and multiple alternatives of different provenance will require additional research in appropriate visual representations; and richer semantic information also expands the possibilities for automating the design of visualizations, beyond statistical graphics.

Acknowledgements

This work was supported in part by an RVAC grant from the Department of Energy and the Max Planck Institute. Thanks to Björn Hartmann for figure assistance and inspiration, and to Mark James for the use of his icons.

References

[1] Lifting the veil: Improving accountability and social transparency in Wikipedia with Wikidashboard.

[2] Virtuoso. http://www.openlinksw.com/.

[3] Linking open data. `http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`, 2008.

[4] S. Auer, C. Bizer, J. Lehmann, G. Kobilarov, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *Proc. ISWC 2007*.

[5] M. Cammarano, X. L. Dong, B. Chan, J. Klingner, J. Talbot, A. Halevey, and P. Hanrahan. Visualization of heterogeneous data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1200–1207, 2007.

[6] A. Doan, R. Ramakrishnan, F. Chen, P. DeRose, Y. Lee, R. McCann, M. Sayyadian, and W. Shen. Community information management. *IEEE Data Eng. Bull.*, 29(1):64–72, 2006.

[7] X. Dong and A. Y. Halevy. Indexing dataspaces. In *Proc. SIGMOD 2007*, pages 43–54, 2007.

[8] M. Dontcheva, S. M. Drucker, D. Salesin, and M. F. Cohen. Relations, cards, and search templates: user-guided web data integration and layout. In *Proc. UIST '07*, pages 61–70, New York, NY, USA, 2007. ACM.

[9] freebase. `http://www.freebase.com`.

[10] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. Proximity search in databases. In *Proc. of VLDB*, 1998.

[11] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proc. of VLDB*, Athens, Greece, 1997.

[12] Google Base. `http://base.google.com/`.

[13] Google Maps. `http://maps.google.com`.

[14] Graphwise. `http://www.graphwise.com`.

[15] J. Heer, F. B. Viégas, and M. Wattenberg. Voyagers and voyeurs: supporting asynchronous collaborative information visualization. In *Proc. CHI '07*, pages 1029–1038, New York, NY, USA, 2007. ACM.

[16] D. F. Huynh, D. R. Karger, and R. C. Miller. Exhibit: lightweight structured data publishing. In *Proc. WWW '07*, pages 737–746, New York, NY, USA, 2007. ACM.

[17] D. F. Huynh, R. C. Miller, and D. R. Karger. Potluck: Semi-ontology alignment for casual users. In *Proc. ISWC 2007*.

[18] Jena. http://jena.sourceforge.net/.

[19] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. In *Proc. VLDB '05*, pages 505–516. VLDB Endowment, 2005.

[20] J. Lehmann, J. Schppel, and S. Auer. Discovering unknown connections - the dbpedia relationship finder. In S. Auer, C. Bizer, C. Mller, and A. V. Zhdanova, editors, *CSSW*, volume 113 of *LNI*, pages 99–110. GI, 2007.

[21] M. Lenzerini. Data integration: a theoretical perspective. In *Proc. PODS '02*, pages 233–246, New York, NY, USA, 2002. ACM.

[22] J. Madhavan, P. A. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. In *ICDE*, pages 57–68, 2005.

[23] J. Madhavan, S. Cohen, X. L. Dong, A. Y. Halevy, S. R. Jeffery, D. Ko, and C. Yu. Web-scale data integration: You can afford to pay as you go. In *CIDR*, pages 342–350, 2007.

[24] MIT. Simile timeline. `http://simile.mit.edu/timeline/`.

[25] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[26] B. Shneiderman, D. Byrd, and W. B. Croft. Clarifying search: A user-interface framework for text searches. *D-Lib Magazine*, 1997.

[27] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proc. WWW '07*, pages 697–706, New York, NY, USA, 2007. ACM.

[28] B. Suh, E. H. Chi, B. A. Pendleton, and A. Kittur. Us vs. them: Understanding social dynamics in wikipedia with revert graph visualizations. *Proc. VAST 2007*, pages 163–170, Oct. 30 2007-Nov. 1 2007.

[29] Swivel. `http://www.swivel.com`.

[30] Tableau. `http://tableausoftware.com`.

[31] F. Viegas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. ManyEyes: a site for visualization at internet scale. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1121–1128, Nov.-Dec. 2007.

[32] W3C. SPARQL Query Language for RDF. `http://www.w3.org/TR/rdf-sparql-query/`.

[33] W3C. The Tabulator. `http://www.w3.org/2005/ajar/tab`.

[34] F. Wu and D. S. Weld. Autonomously semantifying wikipedia. In *Proc. CIKM '07*, pages 41–50, New York, NY, USA, 2007. ACM.