Sources of Evidence for Interactive Table Completion

Jaime Arguello and Robert Capra
School of Information and Library Science, University of North Carolina at Chapel Hill
[jarguello,rcapra]@unc.edu

ABSTRACT

An important question in interactive information retrieval (IIR) is: How can we support searchers with specific types of search tasks? We describe an auxiliary support tool referred to as the "Matrix". The Matrix tool was designed to support searchers with *comparative* search tasks, which require comparing items along different dimensions. The Matrix was designed as a grid of rows and columns representing the items and dimensions related to a comparative task. The Matrix was integrated with a custom-built search interface, which allowed users to search for information and drag-and-drop relevant passages *directly* into cells in the Matrix. We investigate the following general question: Given a partially completed Matrix, can a system automatically populate empty cells in the Matrix with relevant passages? To this end, we conducted two crowdsourced studies in which participants were assigned comparative tasks and asked to use our system (integrated search interface + Matrix) to populate every cell in the Matrix. After gathering this data, we evaluated machine-learned models for ranking passages in response to an *empty* Matrix cell and partially completed Matrix. We address two research questions: (**RQ1**) What are useful types of features for this predictive task? and (**RQ2**) How does performance vary based on the level of Matrix completion? We view our research as a step towards designing support tools that: (1) help users organize information while searching and (2) can *autocomplete* search tasks by exploiting the task structure and a searcher's partial solution.

ACM Reference Format:
Jaime Arguello and Robert Capra. 2020. Sources of Evidence for Interactive Table Completion. In *2020 Conference on Human Information Interaction and Retrieval (CHIIR '20), March 14–18, 2020, Vancouver, BC, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3343413.3377995

1 INTRODUCTION

Tables are a ubiquitous tool for synthesizing information. Tables can be used to support tasks that involve understanding relations between entities or concepts, as well as evaluating alternatives. Broadly speaking, a table is defined by three types of elements: (1) a set of instances (rows); (2) a set of attributes (columns) common to the instances; and (3) a set of attribute-value pairs (cells).

Prior IR research has explored a wide range of *predictive* tasks involving tabular data. For example, prior work has considered ad-hoc table retrieval in response to a keyword query [14] or an input table (i.e., "querying-by-example") [16]. Additionally, prior work has considered table augmentation (i.e., discovering new rows/columns for an input table) [4, 12, 13], population (i.e., populating empty cells) [11], and table generation given a query [15].

In this research, we focus on table *population*—populating a *partially* completed table with relevant information. However, compared to prior work, we focus on table population in the context of a *comparative information-seeking task*. To illustrate, imagine a user interested in buying a new car. This type of task may require comparing alternatives along dimensions that are important to the user, such as resale value, maintenance costs, fuel efficiency, horsepower, etc. To make an informed purchase decision, this user might search for information and generate a table. Using existing tools, this searcher might use a web search engine to find information and an external, *non-integrated* spreadsheet tool to synthesize information in the form of a table. In our research, we envision a system that can provide query-based search and automatic table population in an *integrated* fashion.

To support comparative information-seeking tasks such as the one above, we developed a tool called the "Matrix" (Figure 1). The Matrix was designed as a grid of rows and columns. The rows were designed to correspond to alternatives and the columns were designed to correspond to dimensions associated with the comparative task. The Matrix was integrated into a custom-built search system that allowed users to search for information and drag-and-drop (or copy-and-paste) textual passages from pages found during the search directly into cells in the Matrix.

Integrated note-taking tools such as the Matrix pose important research questions. From a user's perspective, can structured note-taking tools such as the Matrix help searchers complete comparative tasks? For example, can they help searchers keep track of their progress and maintain awareness of knowledge gaps? From a system's perspective, can integrated note-taking tools such as the Matrix help search systems find relevant information? In other words, can a search system use a partially completed Matrix to *automatically* populate the remaining empty cells? If so, what are valuable sources of evidence? Is it evidence derived directly from the textual passages added to the already populated cells? Or is it evidence derived from *user interactions* (e.g., queries, clicks) associated with the already populated cells? Finally, how does autocompletion performance depend on the number of previously populated cells?

In this paper, we investigate algorithms for autocompleting a partially completed Matrix. To this end, we conducted two crowdsourced studies in which participants were assigned comparative tasks and were asked to search for information and complete the Matrix. After gathering this data, we performed experiments using learning-to-rank to autocomplete a partially completed Matrix. We cast this task as a *passage-ranking* task. The input to the system is a partially completed Matrix and a target empty cell, and the goal for the system is to rank passages from the underlying collection that are relevant to the cell. We address two main research questions:

CHIIR '20, March 14–18, 2020, Vancouver, BC, Canada

ACM ISBN 978-1-4503-6892-6/20/03…$15.00
https://doi.org/10.1145/3343413.3377995

- **RQ1:** What types of evidence are most effective for ranking passages for an empty Matrix cell? Using a learning-to-ranking framework, we evaluate different types of features for ranking passages for a given target cell and partially completed Matrix. We compare features derived from other cells in the same *row* vs. other cells in the same *column* as the target empty cell. Additionally, we compare features derived from *passages* added to other cells vs. *search interactions* associated with other cells.
- **RQ2:** How does passage-ranking effectiveness vary based on the number of Matrix cells already populated?

2 RELATED WORK

Prior work has focused on different predictive tasks involving tabular data, including table augmentation (i.e., finding new rows/columns) [4, 11, 13], table population (i.e., populating empty cells) [11], and end-to-end table generation given a keyword query [15].

Discovering new table rows (i.e., entities) is closely related to the task of *entity set expansion*—finding new entities of the same type as a set of input "seeds". Approaches to entity set expansion exploit the fact that related entities appear in similar *contexts*—are surrounded by similar words [3], are embedded in similar markup text (e.g., HTML) [10], have similar attributes in a knowledge base [1], and appear in the same columns in a corpus of tables [9]. Beyond entity expansion, prior work has also focused on attribute expansion (i.e., columns) and table population (i.e., cells). Kopliku et al. [4] focused on attribute expansion given two types of input: a single entity (e.g., U.S.A.) or a set of related entities (e.g., U.S.A., Canada, Mexico). To this end, their approach aimed to discover new attributes using a large table corpus. The system achieved greater performance when given a *set* of entities (vs. only one). Zhang and Balog [13] proposed algorithms for both entity and attribute expansion given an input table. New entities/attributes were discovered using DBPedia and a large table corpus. Results found that new attributes (i.e., columns) could be used to discover new entities (i.e., rows) and vice-versa. Yakout et al. [11] developed a system for augmenting an input table with new columns (along with column headings and populated cells). Their approach used machine learning to match the input table with related tables in a large table corpus. Zhang and Balog [12] developed a system for end-to-end table generation given a keyword query. The task was decomposed into three subtasks: (1) finding entities, (2) finding attributes, and (3) cell population. Subtasks 1 and 2 were performed iteratively, leveraging the current set of entities/attributes to find other entities/attributes. Cell population was performed heuristically. DBPedia and a large table corpus were used to support all three subtasks.

In relation to this prior work, our research focuses on table *population*—finding relevant information for empty cells in an input table. However, compared to prior work, our problem setting is novel in several ways. First, the studies above focused on *entity*-based tables (e.g., Pink Floyd albums) in which the cells contain entities (e.g., record label), numerical values (e.g., release date), or short spans of text describing attribute values (e.g., hit song). In our case, the table cells contain textual *passages* (1-4 sentences) from an underlying document collection. Second, in our setting, table cells can have multiple (even many) relevant passages in the collection. Third, we focus on table population in the context of *interactive search*. Thus, we evaluate the effectiveness of features derived from

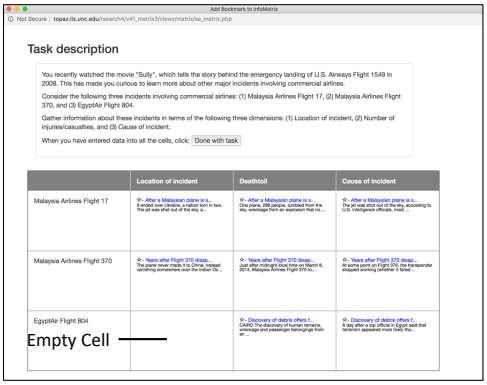

Figure 1: Matrix Tool

textual passages in other cells *as well as* features derived from search interactions that resulted in specific passages being added to other cells. Finally, all the research above used structured data sources (i.e., table corpora and knowledge bases) to inform the cell population task. Using structured data sources influences the nature of the task. For example, cell population boils down to finding the right attribute-value in a knowledge base or the right cell in a table corpus. In our case, the task is to rank passages from an unstructured textual corpus using evidence from the partially completed table.

3 USER STUDIES

Search Interface and Matrix Tool: To investigate algorithms for interactive table completion, we developed a system with two *integrated* components: (1) a custom-build search interface and (2) the Matrix (Figure 1). The Matrix was developed as an experimental tool to support users with comparative search tasks. The Matrix was designed as a table of rows and columns, where the rows are the items being compared and the columns are the dimensions. In our studies (described below), participants were asked to search for information and drag-and-drop relevant passages into cells in the Matrix. Participants could access the Matrix by clicking an "Open Matrix Tool" button on the search interface. Clicking this button displayed the Matrix in a pop-up window. We used Javascript to instrument the drag-and-drop functionality of the system—allowing participants to drag-and-drop a passage from a landing page (rendered by our system) into a Matrix cell. Additionally, we used Javascript to log user interactions on the search interface and Matrix.

Overview of Studies: Two studies (Study 1 and 2) were conducted to gather user interaction data from searchers completing the Matrix for a given task. Both studies were run on Amazon Mechanical Turk (Mturk). Study 1 and 2 had the same protocol but involved different search tasks and document collections. Study 1 used the New York Times (NYT) corpus distributed by the LDC and Study 2 used the TREC Washington Post (WAPO) corpus distributed by NIST. In both studies, participants were assigned comparative tasks that required comparing three items along three dimensions. The *exact* items/dimensions were specified in the task description. For example, one of our tasks asked participants to compare three specific plane crashes/incidents across three dimensions: (1) location of incident, (2) number of injuries/casualties, and (3) cause of incident. We developed 40 tasks (20 per study).

As the main objective of the task, participants were asked to search for information using the custom-built search system and drag-and-drop *at least one* relevant passage into each Matrix cell.

Participants could issue queries and filter search results by year and by topic. The interface retrieved results using Lucene's implementation of the query-likelihood model with Dirichlet smoothing.

Protocol: Study 1 and 2 followed the same protocol. Each Mturk Human Intelligence Task (HIT) involved completing a 3 × 3 Matrix for a comparative task. Before accepting the HIT, participants were provided with instructions and a video introducing the Matrix. Upon accepting the HIT, participants were directed to the main search interface, which included an "Open Matrix Tool" button. Clicking this button displayed a pop-up window with the task description, the (initially empty) Matrix, and a "Done with Task" button. The Matrix contained labeled rows/columns for the three items/dimensions associated with the assigned task. After searching for information and adding passages to every Matrix cell, participants could click the "Done with Task" button to receive a completion code and submit the HIT. Participants were compensated US$3.00 per HIT. Each task was completed by 20 workers, for a total of 800 search sessions (i.e., 2 studies × 20 tasks per study × 20 sessions per task). Tasks were assigned randomly except that participants could not complete the same task more than once.

4 ALGORITHMS AND FEATURES

Given the data gathered for Studies 1 and 2, we evaluated models for ranking passages for a specific empty cell, denoted as (i, j). Conceptually, one could view each target cell (i, j) as a "query" comprised of information from *other* cells in the Matrix. In this respect, each study produced 3,600 "queries" for training and testing (i.e., 20 tasks per study × 20 redundant sessions per task × 9 cells per completed matrix). For the purpose of ranking passages for a target matrix cell (i, j), we used a learning-to-rank framework. We used the implementation of Coordinate Ascent [5] available with the Lemur RankLib toolkit.

We explain our approach and features using the following notation. First, let (i, j) denote the empty cell for which we want to rank passages. Let i and j the cell's row and column. Given (i, j), let $\mathcal{P}_{(i,j)}$ denote the set of candidate passages to rank and p denote a single passage in $\mathcal{P}_{(i,j)}$. Every passage dropped into a matrix cell originated from a specific document. We use $doc(p)$ to denote the document containing p. Similarly, every dropped passage originated from a participant issuing a query and clicking on a search result. We use $Q_{(i,\neg j)}$ and $Q_{(\neg i,j)}$ to denote the set of queries resulting in passages added to *other* cells in row i and *other* cells in column j. Similarity, we use $C_{(i,\neg j)}$ and $C_{(\neg i,j)}$ to denote the set of clicked snippets resulting in passages added to *other* cells in row i and *other* cells in column j. We generated three categories of features. Our features assume that passages relevant to (i, j) are similar or related to passages added to *other* cells in row i and column j.

Label Features: We generated 6 features from the row and column labels associated with cell (i, j). To generate these features, we treated the row/column labels as queries and ranked passages/documents using the query-likelihood model.

(1) Given the label for row i, the reciprocal rank of p.
(2) Given the label for column j, the reciprocal rank of p.
(3) Given the concatenated labels for i and j, the reciprocal rank of p.
(4) Given the label for row i, the reciprocal rank of $doc(p)$.
(5) Given the label for column j, the reciprocal rank of $doc(p)$.
(6) Given the concatenated labels for i and j, the reciprocal rank of $doc(p)$.

Passage Features: We generated 34 features from passages added to *other* cells in row i and *other* cells in column j. For each of the first 8 items below, we generated 4 features by considering the min, max, mean, and st. dev. of comparisons with other passages in the same row/column. To compute the GloVe cosine similarity features, we used the pre-trained GloVe word embeddings [7] and represented each passage/snippet using its TF.IDF-weighted *average* word embedding [6].

(1) TF.IDF cosine similarity between p and other passages in row i.
(2) TF.IDF cosine similarity between p and other passages in column j.
(3) GloVe cosine similarity between p and other passages in row i.
(4) GloVe cosine similarity between p and other passages in column j.
(5) Jaccard similarity between categories associated with $doc(p)$ and documents associated with other passages in row i.
(6) Jaccard similarity between categories associated with $doc(p)$ and documents associated with other passages in column j.
(7) Publication year difference between $doc(p)$ and documents associated with other passages in row i.
(8) Publication year difference between $doc(p)$ and documents associated with other passages in column j.
(9) Number of passages in row i that also originated from $doc(p)$.
(10) Number of passages in column j that also originated from $doc(p)$.

Interaction Features: We generated 32 user interaction features that considered the queries and clicks that resulted in passages being added to *other* cells in row i and column j. For each of the 8 items below, we generated 4 features by considering the min, max, mean, and st. dev. of comparisons with queries/clicks that resulted in passages dropped into *other* cells in same row/column.

(1) Given queries in $Q_{(i,\neg j)}$, reciprocal rank of p.
(2) Given queries in $Q_{(\neg i,j)}$, reciprocal rank of p.
(3) Given queries in $Q_{(i,\neg j)}$, reciprocal rank of $doc(p)$.
(4) Given queries in $Q_{(\neg i,j)}$, reciprocal rank of $doc(p)$.
(5) TF.IDF cosine similarity between p and snippets in $C_{(i,\neg j)}$.
(6) TF.IDF cosine similarity between p and snippets in $C_{(\neg i,j)}$.
(7) GloVe cosine similarity between p and snippets in $C_{(i,\neg j)}$.
(8) GloVe cosine similarity between p and snippets in $C_{(\neg i,j)}$.

5 EVALUATION METHODOLOGY

We trained and tested models to rank passages for a given matrix cell (i, j). Relevance labels for training/testing were derived from passages added to cell (i, j) by any of the 20 participants who completed the same task. All passages added to (i, j) were considered *relevant* and all others *not relevant*. To generate candidate passages $\mathcal{P}_{(i,j)}$ for (i, j), we constructed a "query" by concatenating the row/column labels and included all passages from the top-100 documents. Models were trained/tested using 20-fold cross-validation (separately for the NYT and WAPO testbeds). To test the generalizability of models to *new* tasks, each training fold included data from 19 tasks and each test fold included data for the held-out task.

6 RESULTS

Table 1 shows NDCG@30 performance for both evaluation testbeds, Study 1 (NYT corpus and tasks) and Study 2 (WAPO corpus and tasks). The columns in Table 1 correspond to models trained using different sets of features (RQ1). Column ALL indicates performance for models trained using all features. The rows correspond to models trained under different levels of Matrix completion (RQ2), from 8 populated cells (only (i, j) empty) to only 1 populated cell. The percentage values indicate the percentage drop in performance

compared to ALL for the same row (i.e., same level of Matrix completion). Statistical significance was tested using the randomization test [8]. Symbol '$\triangledown$' denotes a statistically significant difference compared to ALL after Bonferroni correction. Given our large dataset (3,600 target cells (i, j) per testbed), most differences were significant (even after correction). Thus, we focus our attention on large performance differences. With respect to RQ1 and RQ2, results on both testbeds found similar trends.

RQ1: To investigate RQ1, we conducted two different feature ablation analyses. Our approach to passage-ranking assumes that relevant passages for cell (i, j) are similar (or related) to other passages in row i and column j. Thus, our first ablation analysis compares models trained using only features derived from the same row (ONLY.ROW) versus only features derived from the same column (ONLY.COL). Our results suggest two main trends. First, both groups of features added value. Ignoring row features (ONLY.COL) and ignoring column features (ONLY.ROW) both resulted large drops in performance compared to ALL. Second, row features were much more predictive than column features (ONLY.ROW $\gg$ ONLY.COL). Recall that rows corresponded to items and columns to dimensions.

Our second ablation analysis compares models using different groups of features: label, passage, and interaction features. Our results suggest three main trends. First, using label features alone (ONLY.LABEL) produced the worst performance (ALL $\gg$ ONLY.LABEL). Second, passage features contributed more to performance than interaction features (LABEL+PASS > LABEL+INTER). Finally, interaction features provided some value (LABEL+INTER > ONLY.LABEL), but *not* when we also included passage features (ALL $\approx$ LABEL+PASS). In other words, interaction features did not contribute valuable evidence that was *complementary* to passage features.

RQ2: Table 1 shows performance for different levels of table completion. Our RQ2 results suggest three main trends. As expected, passage-ranking performance improves as the Matrix is more complete. Second, this trend was observed for all models using features derived from Matrix cells (ALL, ONLY.ROW, ONLY.COL, LABEL+PASS, LABEL+INTER). The same trend was not observed for label features because they do not rely on other cells being populated. Slight variations for ONLY.LABEL are due to the random parameter initialization in the Coordinate Ascent algorithm. Third, as the Matrix is completed, performance increases at a *slower* rate (i.e., diminishing marginal returns). To illustrate, across all feature-combinations, the difference in performance between 1 and 2 populated cells is greater than between 7 and 8 populated cells.

Table 1: Passage-ranking performance (NDCG@30).

cells pop.	ALL	ONLY.ROW (no col)	ONLY.COL (no row)	ONLY.LABELS (no pass/inter)	LABELS+INTER (no pass)	LABELS+PASS (no inter)
8	.299	.230 (-23.1%)$^{\triangledown}$	.100 (-66.6%)$^{\triangledown}$	.197 (-34.1%)$^{\triangledown}$	.240 (-19.7%)$^{\triangledown}$	.296 (-1.0%)
7	.290	.223 (-23.1%)$^{\triangledown}$	.098 (-66.2%)$^{\triangledown}$	.200 (-31.0%)$^{\triangledown}$	.238 (-17.9%)$^{\triangledown}$	.287 (-1.0%)
6	.281	.217 (-22.8%)$^{\triangledown}$	.092 (-67.3%)$^{\triangledown}$	.199 (-29.2%)$^{\triangledown}$	.234 (-16.7%)$^{\triangledown}$	.278 (-1.1%)
5	.272	.210 (-22.8%)$^{\triangledown}$	.082 (-69.9%)$^{\triangledown}$	.198 (-27.2%)$^{\triangledown}$	.232 (-14.7%)$^{\triangledown}$	.269 (-1.1%)
4	.261	.205 (-21.5%)$^{\triangledown}$	.071 (-72.8%)$^{\triangledown}$	.198 (-24.1%)$^{\triangledown}$	.225 (-13.8%)$^{\triangledown}$	.260 (-0.4%)
3	.247	.194 (-21.5%)$^{\triangledown}$	.057 (-76.9%)$^{\triangledown}$	.197 (-20.2%)$^{\triangledown}$	.219 (-11.3%)$^{\triangledown}$	.243 (-1.6%)$^{\triangledown}$
2	.233	.188 (-19.3%)$^{\triangledown}$	.042 (-82.0%)$^{\triangledown}$	.200 (-14.2%)$^{\triangledown}$	.214 (-8.2%)$^{\triangledown}$	.235 (0.9%)
1	.217	.175 (-19.4%)$^{\triangledown}$	.025 (-88.5%)$^{\triangledown}$	.198 (-8.8%)$^{\triangledown}$	.204 (-6.0%)$^{\triangledown}$	.214 (-1.4%)

(a) NYT Results

cells pop.	ALL	ONLY.ROW (no col)	ONLY.COL (no row)	ONLY.LABELS (no pass/inter)	LABELS+INTER (no pass)	LABELS+PASS (no inter)
8	.345	.294 (-14.8%)$^{\triangledown}$	.095 (-72.5%)$^{\triangledown}$	.227 (-34.2%)$^{\triangledown}$	.293 (-15.1%)$^{\triangledown}$	.339 (-1.7%)$^{\triangledown}$
7	.337	.288 (-14.5%)$^{\triangledown}$	.091 (-73.0%)$^{\triangledown}$	.227 (-32.6%)$^{\triangledown}$	.287 (-14.8%)$^{\triangledown}$	.331 (-1.8%)$^{\triangledown}$
6	.329	.281 (-14.6%)$^{\triangledown}$	.086 (-73.9%)$^{\triangledown}$	.229 (-30.4%)$^{\triangledown}$	.282 (-14.3%)$^{\triangledown}$	.323 (-1.8%)$^{\triangledown}$
5	.318	.272 (-14.5%)$^{\triangledown}$	.079 (-75.2%)$^{\triangledown}$	.226 (-28.9%)$^{\triangledown}$	.279 (-12.3%)$^{\triangledown}$	.315 (-0.9%)
4	.305	.261 (-14.4%)$^{\triangledown}$	.068 (-77.7%)$^{\triangledown}$	.227 (-25.6%)$^{\triangledown}$	.274 (-10.2%)$^{\triangledown}$	.302 (-1.0%)
3	.293	.249 (-15.0%)$^{\triangledown}$	.062 (-78.8%)$^{\triangledown}$	.229 (-21.8%)$^{\triangledown}$	.268 (-8.5%)$^{\triangledown}$	.288 (-1.7%)$^{\triangledown}$
2	.278	.235 (-15.5%)$^{\triangledown}$	.049 (-82.4%)$^{\triangledown}$	.227 (-18.3%)$^{\triangledown}$	.261 (-6.1%)$^{\triangledown}$	.279 (0.4%)
1	.259	.223 (-13.9%)$^{\triangledown}$	.035 (-86.5%)$^{\triangledown}$	.235 (-9.3%)$^{\triangledown}$	.245 (-5.4%)$^{\triangledown}$	.257 (-0.8%)$^{\triangledown}$

(b) WAPO Results

7 DISCUSSION AND CONCLUSION

Our results have implications for designing tools such as the Matrix.

Row vs. Column Features: Our passage-ranking approach assumes that passages relevant to an empty cell are similar (or related) to passages added to other cells in the same row (item) and column (dimension). Our RQ1 results validate this assumption (ALL > ONLY.ROW, ONLY.COL). Thus, Passage-ranking algorithms should harness evidence from the same row/column as an empty cell.

While both feature types helped, row features were more predictive than column features (ONLY.ROW $\gg$ ONLY.COL). We see two possible explanations for this trend. In our case, items were typically concrete nouns with specific names (e.g., Hurricane Katrina). Conversely, many of our dimensions were abstract concepts with more varied language (e.g., government response). Results from prior research also suggest that searching for specific items is easier than specific dimensions [2]. A second possibility is that documents tend to focus on items (e.g., entities or events) rather than dimensions. This is certainly the case for news articles.

Passage vs. Interaction Features: Our RQ1 results suggest that both passage and interaction features provided valuable evidence for ranking passages (LABELS+PASS, LABELS+INTER > LABELS.ONLY). This trend suggests that passages relevant to an empty cell are not only similar to other passages in the same row/column in a "text similarity" sense. They were also found by participants using similar search interactions (e.g., similar queries and clicked results). While both feature types helped, interaction features *did not* contribute evidence that was *complementary* to passage features (ALL $\approx$LABEL+PASS). Future work will consider *other* types of interaction features that may provide complementary evidence.

Effects of Matrix Completion: As expected, passage-ranking performance improved as the Matrix was more complete (RQ2). This result suggests that interactive table population systems such as the Matrix should be more confident as more data is added to the table. Interestingly, as the Matrix was more complete, performance improved with slightly diminishing marginal returns.

Concluding Remarks: The Matrix tool was developed to help users organize information during comparative tasks. It is an example of a structured note-taking tool that can be integrated into a search system to assist users with specific task types. Our results suggest that systems may be able to exploit the structure of a task to *autocomplete* a user's partial solution.

While our results are encouraging, open questions remain. In future work, we will evaluate the Matrix tool from a searcher's perspective. For example, does the Matrix help searchers track their progress and maintain awareness of gaps in knowledge? Future work may also consider other tools designed to autocomplete tasks with different types of structure. For example, imagine a tool that helps searchers organize information into clusters during exploratory search tasks. Such a tool may be able to use the information in the current set of clusters (and associated interaction data) to retrieve related documents/passages.

Acknowledgements: This research was supported by NSF grants IIS-1552587 and IIS-1451668.

REFERENCES

[1] Marc Bron, Krisztian Balog, and Maarten de Rijke. 2013. Example Based Entity Search in the Web of Data. In *ECIR*. Springer Berlin Heidelberg, 392–403.

[2] Robert Capra, Jaime Arguello, Heather O'Brien, Yuan Li, and Bogeum Choi. 2018. The Effects of Manipulating Task Determinability on Search Behaviors and Outcomes. In *SIGIR*. ACM, 445–454.

[3] Yeye He and Dong Xin. 2011. SEISA: Set Expansion by Iterative Similarity Aggregation. In *WWW*. ACM, 427–436.

[4] Arlind Kopliku, Mohand Boughanem, and Karen Pinel-Sauvagnat. 2011. Towards a Framework for Attribute Retrieval. In *CIKM*. ACM, 515–524.

[5] Donald Metzler and W. Bruce Croft. 2007. Linear Feature-based Models for Information Retrieval. *Information Retrieval* 10, 3 (2007), 257–274.

[6] Bhaskar Mitra and Nick Craswell. 2018. An Introduction to Neural Information Retrieval. *Foundations and Trends in Information Retrieval* 13 (2018), 1–126.

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[8] Mark D. Smucker, James Allan, and Ben Carterette. 2007. A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. In *CIKM*. ACM, 623–632.

[9] Chi Wang, Kaushik Chakrabarti, Yeye He, Kris Ganjam, Zhimin Chen, and Philip A. Bernstein. 2015. Concept Expansion Using Web Tables. In *WWW*. IW3C2, 1198–1208.

[10] Richard C. Wang and William W. Cohen. 2007. Language-Independent Set Expansion of Named Entities Using the Web. In *ICDM*. IEEE Computer Society, 342–350.

[11] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. 2012. InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables. In *SIGMOD*. ACM, 97–108.

[12] Shuo Zhang. 2018. SmartTable: Equipping Spreadsheets with Intelligent Assistance Functionalities. In *SIGIR*. ACM, New York, NY, USA, 1447–1447.

[13] Shuo Zhang and Krisztian Balog. 2017. EntiTables: Smart Assistance for Entity-Focused Tables. In *SIGIR*. ACM, 255–264.

[14] Shuo Zhang and Krisztian Balog. 2018. Ad Hoc Table Retrieval Using Semantic Similarity. In *WWW*. International World Wide Web Conferences Steering Committee, 1553–1562.

[15] Shuo Zhang and Krisztian Balog. 2018. On-the-fly Table Generation. In *SIGIR*. ACM, 595–604.

[16] Shuo Zhang and Krisztian Balog. 2019. Recommending Related Tables. *CoRR* abs/1907.03595 (2019). http://arxiv.org/abs/1907.03595