UberTagger: Tagging Micro-interactions for Hypothesis Generation

1st Author Name
Affiliation
Address
e-mail address
Optional phone number

2nd Author Name
Affiliation
Address
e-mail address
Optional phone number

3rd Author Name
Affiliation
Address
e-mail address
Optional phone number

ABSTRACT

As interactive systems become more complex, interaction designers may have more difficulty generating clear hypotheses to explain issues hindering user success in performing tasks. As with other debugging tasks, visualization and machine learning systems have been helpful but hypothesis generation is still challenging.

We take an intelligent user interface approach and present UberTagger for exploratory analysis of micro-interaction data that combines annotation and visual analytics, assisted by a recommender system. UberTagger is designed to help interaction designers explore large datasets of mouse and keyboard input captured in crowdsourcing experiments. We implement a novel tagging system to support the data transformations specified in the principles of Exploratory Sequential Data Analysis. Tags can label different media types (e.g., subsets of a time-series or cells in a table) but exist in a common searchable environment. This approach further supports meta-tags, tag structures and hierarchies, and meta-analysis of the annotation process. This unique tagging approach, for data analysis, annotation, and algorithms, can help with the detection and design of micro-interactions for hypothesis generation.

Author Keywords
Micro-interaction; Hypothesis Generation; Recommender System; Tagging; Crowdsourcing; Exploratory Sequential Data Analysis.

ACM Classification Keywords
H.5.2. [Information interfaces and presentation]: User Interfaces – User Interfaces

INTRODUCTION

Successful interaction design makes users faster, more accurate, or more satisfied with their efforts and outcomes. However, as interactive systems become more complex, interaction designers may have more difficulty in creating successful interactions [10]. When user success is lower than expected, it may be challenging for designers to generate hypotheses to explain issues users experienced [17].

Hypothesis generation results in potential explanations of a set of observations. The challenges of generating hypotheses have been discussed [5] leading to strategies [8] including situational logic (akin to debugging where the unique logic of the current situation is used to help identify logical antecedents or consequences to develop a plausible scenario), applying theory, comparison with previous situations, and data immersion. We adopt a data immersion strategy guiding hypothesis development by observations collected from crowdsourcing experiments.

The practice of exploratory data analysis, as proposed by Tukey [35], is "detective work" to "uncover indications" supporting hypothesis generation. Within the area of usability [9], a set of principles called Exploratory Sequential Data Analysis (ESDA) [29] proposed eight data transformations needed to support a hypothesis generation workflow. Micro-interaction data [13,21,26,42] has been proposed as a means of designing, finding, and fixing subtle interaction problems. Several systems have been developed for visual analysis of complex datasets [34,37,39] including systems focused on usability and micro-interaction data [3,6,20,30]. These systems go beyond event logging [1,24] but do not provide a single evaluation environment for the "composition of various transformations, analyses, and visualizations" [12].

To address these challenges we present *UberTagger* for data-driven hypothesis generation based on the exploratory analysis of micro-interaction data. UberTagger combines annotation and analysis, assisted by a recommender system, to help interaction designers explore large datasets of mouse and keyboard input captured in crowdsourcing experiments, using the overarching concept of tagging [16].

As an open source project, we hope UberTagger will foster exploratory data analysis and tool creation for hypothesis generation.

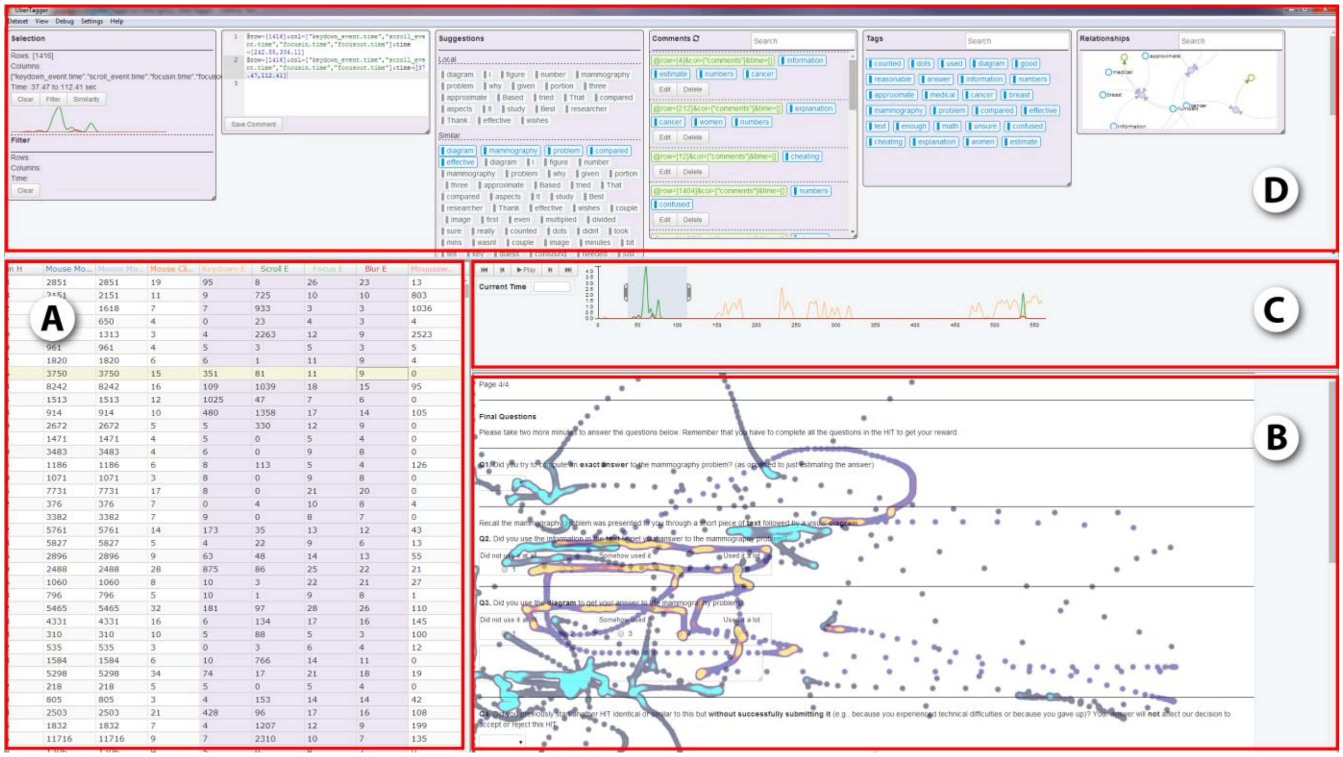


Figure 1. The UberTagger user interface. Top (D): dashboard including tools for Selection, Comments, Suggestions, Tags, and Relationships. Bottom-left (A): observations data grid. Bottom-right: (C) temporal and (B) spatial visualizations.

UBERTAGGER

We first describe the data sources and their display in UberTagger followed by the conceptual design and a discussion on how the user interface supports this design. We then review the ESDA data transformations and how they appear in the system and, finally, describe how the embedded recommender system is designed and implemented.

Data

UberTagger is a desktop application (Node-WebKit [38]) and can load data collected from crowdsourcing experiments [3]. Simple JSON documents, or TingoDB [11], a simple embedded JavaScript database, may be loaded. The use of an embedded database removes the requirement for external dependencies and complicated set-up.

For a dataset to be used in UberTagger, it requires that package.json [31] is in the main folder of the dataset, and includes a schema for how to access the data (whether the data is in MongoDB, TingoDB, or a JSON file), what fields are in the data, and how to visualize the data (which data fields should be input for which views). package.json is a package definition for node.js modules, and we extend it to add UberTagger functionality.

User Interface

The user interface of UberTagger consists of four panels (see Figure 1). The bottom-left panel (Figure 1A) is the Observations panel. In the examples in this paper, each row in this data grid panel represents the data collected from a specific participant of a crowdsourcing experiment. The columns are the types of data collected, such as user ID, screen size, comments, as well as aggregates such as number of mouse clicks, scroll units, etc.

The bottom-right (Figure 1B) Visualization panel is shows what the user-interface appeared like at a specific point in time depending on what events were given to it. Overlaid on the webpage context from the experiment, this panel can display a heatmap of mouse movement.

Above the Visualization panel, the top-right (Figure 1C) Timeline panel shows (in this case) x and y mouse positions over time. An interval of time is currently selected highlighting the corresponding part of the heatmap in yellow.

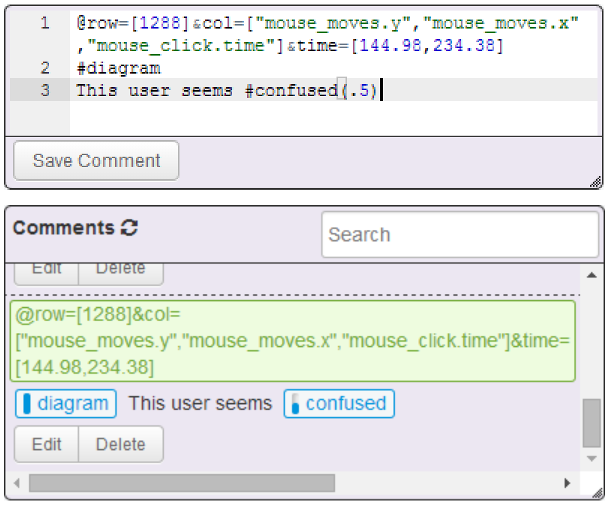
The top full-width (Figure 1D) panel is the dashboard area containing floating tools indicating the relevant Selection, Comments, Tags, Relationships, and Suggestions (similar data fragments recommended for automated tagging).

Tagging System

Tags are lightweight annotations popularized by social media systems such as Twitter, often comprised of single words denoted by a preceding hash sign: "#". They can appear alone or embedded in text, for example, a comment may include a tag: "`This participant may be #confused.`" In addition to this practice, we also adopt "@" to indicate an address which, in UberTagger, specifies a location in the data set. For example, the comment above could be made more specific: “This @user=[27] may be #confused.” In addition to our usage of the typical “#” and “@” we include a mechanism to specify a simple parent-child tag relationship using “/”. For example, “#cognitive/confused” indicates that the “confused” tag is a child of a tag category called “cognitive”. Finally, to support users in ranking relevance, tags may have optional weights, for example “#confused(0.2)”.

Comments

Comments are the only artifacts made by users in UberTagger. They are time-stamped and unique in the system, even if the contents of multiple comments are identical. Using the tagging system described above, comments can contain any number of tags and any number of selections. The comment creation panel (see Figure 2, top) shows selected rows and columns of the Observations table and a subset of the time series of mouse position input data. It also shows a #diagram tag and a weighted embedded tag. When the comment is saved, it appears in the Comments panel which lists all the comments saved for this data set (see Figure 2, bottom).



```
@row=[1288]&col=["mouse_moves.y","mouse_moves.x"
,"mouse_click.time"]&time=[144.98,234.38]
#diagram
This user seems #confused(.5)
```

@row=[1288]&col=["mouse_moves.y","mouse_moves.x","mouse_click.time"]&time=[144.98,234.38]

diagram This user seems confused

Figure 2. (top) The comment creation panel with selection, free-text of a tag, and a remark with an embedded weighted tag. (bottom) Same comment saved to Comments list panel.

Relationships

To help reveal patterns of re-use, we show the three primary entities in UberTagger, comments, tags, and selections, in the Relationships panel (see Figure 3). In general, these elements form a number of directed cyclic graphs.

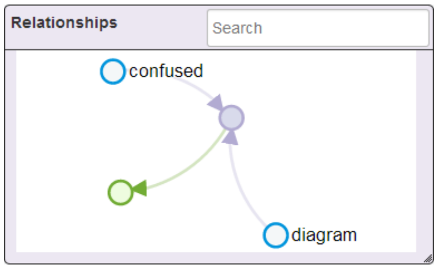


Figure 3. The Relationship panel showing tags (blue), selections (green) and comments (purple).

In the CommentSpace system [40] tags and links are also a central concept and support hypothesis generation but only a small, fixed vocabulary of tags (*question*, *hypothesis*, *to-do*) and links (*evidencefor*, *evidence-against*) are used.

Having discussed the primary concepts and their use in the interface, we review the ESDA data transformations together with the interface panels and elements that support them.

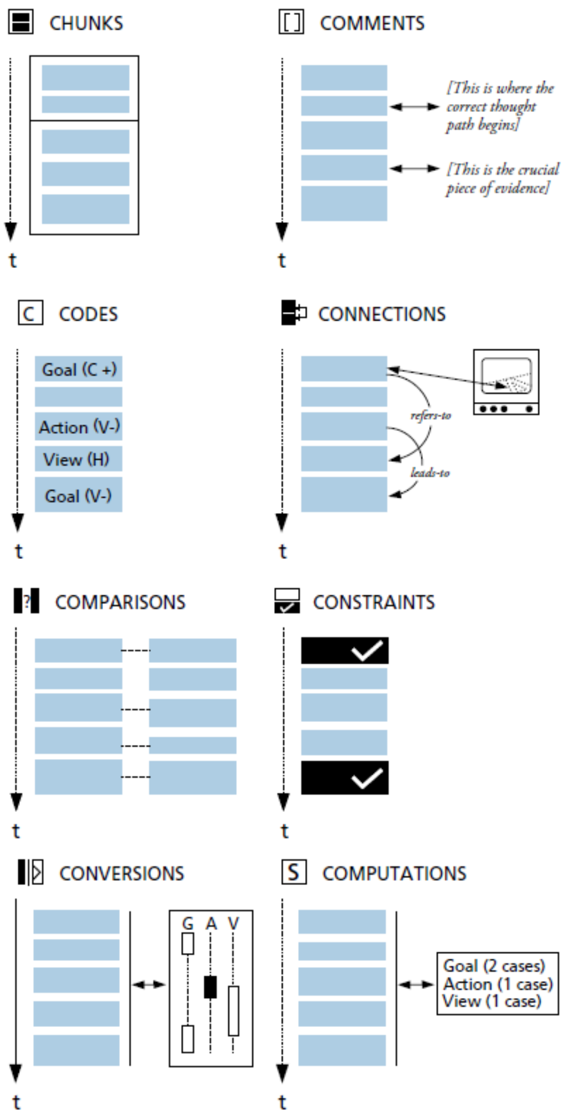


Figure 4: The “Eight C” data transformations in ESDA [29].

DATA TRANSFORMATIONS – ESDA

As mentioned earlier, hypothesis generation is the outcome of a process of exploratory sequential data analysis (ESDA). In the example used throughout this paper, this refers to tagging sequential data (time series of micro-interactions), in the context of its underlying webpage, together with sets of observations between participants. To guide the design of features needed for this level of analysis, we follow the guidelines of Sanderson and Fisher [4,29] called ESDA, originally proposed for video analysis of human-computer interaction (HCI) tasks. This work introduced eight fundamental transformations (see Figure 4), which can be performed on or within sequential data, as being critical for scientific inference in ESDA. The transformations can be grouped as being for the purpose of *Annotation* (Chunks, Comments, Codes, Connections) or *Analysis* (Comparisons, Constraints, Conversions, Computations). The inputs to these transformations are any selectable items in the UberTagger interface.

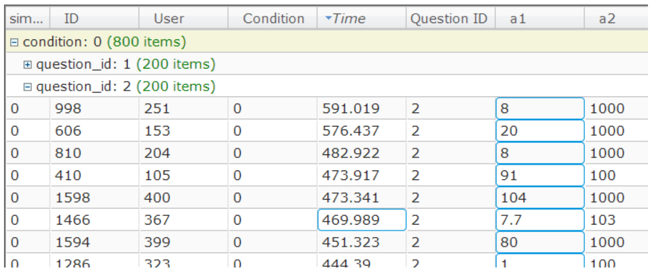
Using the traditional selection-action interaction model, we adhere to a global unifying principle that anything in the interface that is selectable is also taggable. Tagged items or areas become the atomic units of input to the Annotation and Analysis transformation functions. The output of these transformations are themselves selectable and may be transformed supporting meta-annotation.

We next describe the eight transformations, showing how they operate in UberTagger, grouped as Annotations and Analysis.

Annotation Transformations

Chunks

Chunks are “segments of adjacent data elements that the analyst perceives as forming a coherent group” [4]. In UberTagger for example, rows in the observations data grid may be sorted by a text-field and a group of rows that all have the same value in the text-field will become adjacent and may be selected as a group and tagged. Adjacent columns may be grouped as a chunk or a selected time range of values may be tagged as a chunk. In additional to manual grouping, some automatic grouping is provided, by clicking the column header, to avoid the need to tag logical groups of observations, for example, grouping by condition or question (see Figure 5). Also, for any cells or cell groups that have already been tagged, a tag style is applied so users can see that comments have already been made about them and hovering will provide a tooltip with details.



sim...	ID	User	Condition	Time	Question ID	a1	a2
condition: 0 (800 items)							
question_id: 1 (200 items)							
question_id: 2 (200 items)							
0	998	251	0	591.019	2	8	1000
0	606	153	0	576.437	2	20	1000
0	810	204	0	482.922	2	8	1000
0	410	105	0	473.917	2	91	100
0	1598	400	0	473.341	2	104	1000
0	1466	367	0	469.989	2	7.7	103
0	1594	399	0	451.323	2	80	1000
0	1286	323	0	444.39	2	1	100

Figure 5: Automatic grouping and tag indicators.

Comments

Comments are “unstructured informal or formal notes that the analyst attaches to data elements, to chunks, or even to the results of intermediate analyses” [4]. As can be seen in Figure 2, comments may be attached to any selection. As mentioned above, comments may also introduce new tags or reference existing tags. In UberTagger, comments can have a single selection or multiple selections, and may contain a single hash-tag name or free-text notes with embedded tags or references. As will be discussed later, comments can also call JavaScript functions including plotting functions.

Codes

Codes are simple user-defined tags “attached to data elements or chunks designed to capture the meaning of the data while reducing the variability of its vocabulary” [4]. In UberTagger, tags and codes are synonymous. Once a selection is tagged, a number of operations can be performed relative to a given tag, to be discussed in detail in a later section (see Figure 6).

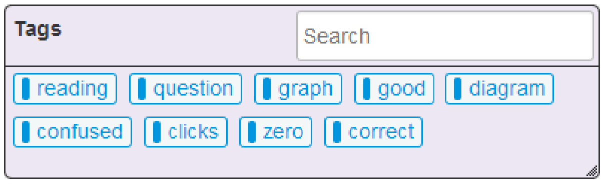


Figure 6. The Tag panel shows the tags in the current comment or all tags when no comment is selected.

Connections

Connections are “a means of following threads through their nonlinear paths and identifying the relationship among their elements” [4]. To link selections of similar or dissimilar types, the tag hierarchy mechanism can be used. For example, a selection can be tagged as #tag/subtag1 while a second selection is tagged as #subtag1/subtag2. When shown in the Relationships tool, the linear tag list will be shown. In practice, users can refer back to the connections to help re-use tags and build tag hierarchies (see Figure 7).

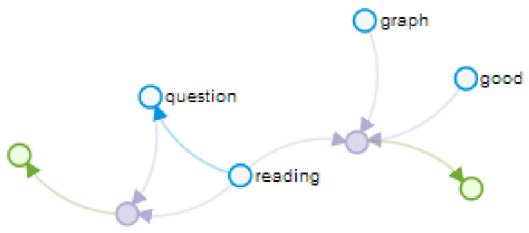


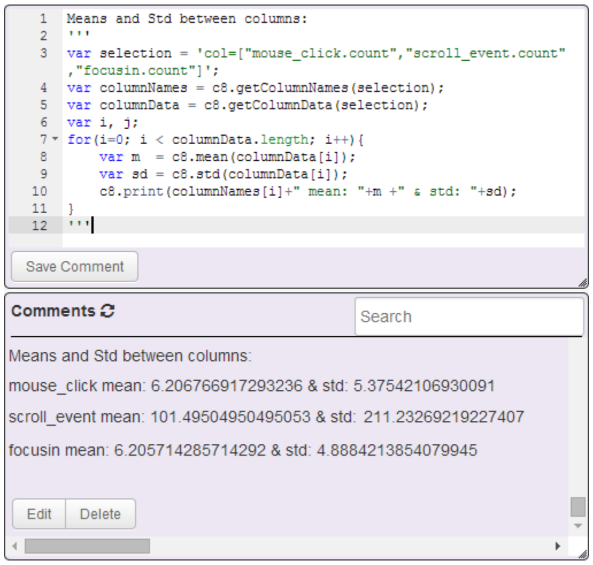
Figure 7. The Relationships panel shows how tags, comments, and selections are related.

Analysis Transformations

Computations

Computations “reduce the data to summary representations, including simple counts, complex quantitative relationships, or tests of statistical significance” [4]. We add the ability to perform computation through a JavaScript API accessible through the commenting system. Three apostrophes indicate the beginning and ending of a code block. When the code block is added, using the “Save Comment” button, the computation is performed and the output is rendered as part of the comment. For example, the user could add code to run a number of correlations (see Figure 8). To re-evaluate the computation, the user can edit the comment and re-save it or click a global refresh button next to Comments header (See Figure 2 bottom), which is useful if the input data changes.

We use exiting open source statistics and math libraries such as statkit [25], jstat [41], NumericJS [14], and the D^3 toolkit for plotting [2]. Note that our implementation does not currently offer any debugging functionality or protect the user from programming errors that may crash the application, create infinite loops, etc.



```
Means and Std between columns:
'''
var selection = 'col=["mouse_click.count","scroll_event.count"
,"focusin.count"]';
var columnNames = c8.getColumnNames(selection);
var columnData = c8.getColumnData(selection);
var i, j;
for(i=0; i < columnData.length; i++){
    var m  = c8.mean(columnData[i]);
    var sd = c8.std(columnData[i]);
    c8.print(columnNames[i]+" mean: "+m +" & std: "+sd);
}
'''
```

```
Means and Std between columns:
mouse_click mean: 6.206766917293236 & std: 5.37542106930091
scroll_event mean: 101.49504950495053 & std: 211.23269219227407
focusin mean: 6.205714285714292 & std: 4.8884213854079945
```

Figure 8. The Comments panel is a general JavaScript editor. Shown here is code for the evaluation of correlations between columns.

Comparisons

Comparisons "demonstrate the effects of different treatments of the data with one another" [4]. For example, one might compare time spent in a specific region against total mouse travel distance in that region as a measure of usefulness or confusion. For visual comparisons, multiple selections of observations and columns overlays multiple plots in the timeline on the same axis (see Figure 9). Sorting also gives ability to rank items relative to each other and to see the rows of observations adjacent to one another. Sorting by similarity allows for ranking based on multiple features, variables, or columns.

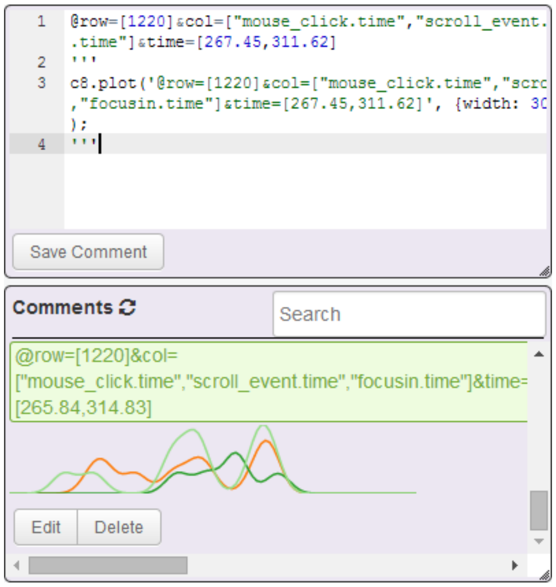
The computation JavaScript API also allows for comparisons to be performed on a quantitative statistical level using basic statistical functions such as mean, standard deviation, etc. Calculations such as correlations and ANOVA add a statistical means to compare different variables in the observations as well. In UberTagger, Computation supports several types of Comparison.

Constraints

Constraints are applied to data to filter out items or to select specific items. For example, all of the participants who entered the same answer could be selected using a filter. Or a filter could be added to remove alpha key events from numeric entry type-in fields. In UberTagger, several dashboard panels will automatically filter their contents based on the current selection. Sets of filtered items could be tagged for further transformation.

Conversions

Conversions "transform data in order to reveal new patterns" [4]. This includes converting the mouse data to a heatmap, as shown in Figure 1B or plotting a number of event types over a time interval (see Figure 9). Our computation API provides basic plots such as line graphs, histograms, box-plots, etc.



```
@row=[1220]&col=["mouse_click.time","scroll_event.
.time"]&time=[267.45,311.62]
'''
c8.plot('@row=[1220]&col=["mouse_click.time","scrc
,"focusin.time"]&time=[267.45,311.62]', {width: 3C
);
'''
```

```
@row=[1220]&col=
["mouse_click.time","scroll_event.time","focusin.time"]&time=
[265.84,314.83]
```

Figure 9. The Comments panel may be used to arbitrarily transform data. Shown here is a plot of a time interval of various events for a specific row (user ID).

RECOMMENDER SYSTEM

The automatic recommendation of tags can help users to re-use and organize tags. The Suggestions panel, as shown in Figure 10, dynamically updates and displays recommended tags based on the current selection. Tags colored blue already exist in the data set as user-defined tags. Auto-tags are shown in grey and are system-generated using three algorithms: local, similar, and co-occurrences. The auto-tags are tokens parsed from the comments using a Natural Language Processing library [36] with Part-of-Speech information extracted using the WordNet [19] database.

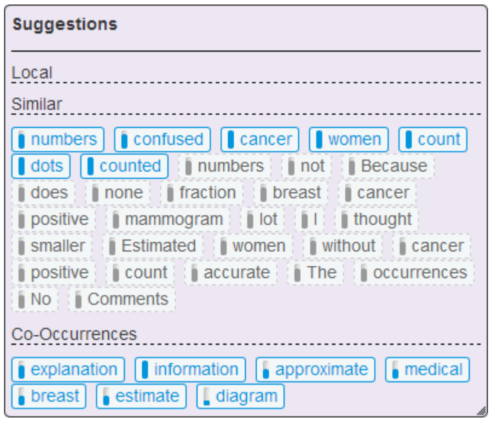


Figure 10. The Suggestions panel shows recommended tags based on text in the comment (empty in this case), similar tags and comments from the full dataset, and tag co-occurrences.

Auto-tags

Local
Any rows in the grid that are currently selected are parsed for any text fields, and all the auto-tags are extracted, tokenized, filtered, and displayed.

Similar
Based on the current selection, the most similar rows are found by looking through all the text fields in the record and associated comments, and are parsed for auto-tags. Any tags that are associated with those similar items are shown, up to a threshold to limit the number of records returned (i.e. 10). The similarity method is described in the Similarity Measure section below.

Co-occurrences
All tags that were discovered through the similarity search, discussed below, also undergo an extra co-occurrence stage, where all the comments are parsed for any other tags that may occur together with the tags already discovered. As the user types in the comment panel and types in new tags, the system parses the text input after each keystroke and updates the Suggestion panel with any existing tags that co-occur with the newly added one.

"Progress-bar" Indicator
For both tags and auto-tags in the Similar and Co-occurrence section of the Suggestions panel, the tags vertical "progress bar" shows the similarity associated with the most similar record that has a connection to this tag (see Figure 10). The actual numerical similarity value is displayed in a tooltip when the cursor hovers over the tag. In the Co-occurrence section, the tags "progress bar" indicates the number of co-occurrences in the existing comments. As the user clicks on the tags, they are inserted into the current comment.

Similarity Measure
In order to give suggestions for possible tags for an untagged selection in the data, we can find similar items in the dataset that have already been tagged. To find similarity between different observations in the dataset we combine multiple similarity metrics that operate on different data types. We support similarity of numeric, textual, and time series data.

The overall similarity of the given selection is calculated by combining the different individual similarities by a simple linear combination:

$$\text{similarity}(\text{selection}) = s_n * \mathrm{w}_n + s_t * \mathrm{w}_t + s_s * \mathrm{w}_s,$$

where
s_n is similarity of numeric data from observation table,
w_n is weight of numeric term,
s_t is the similarity of textual data (from participant comments in our sample data set),
w_t is the weight of the textual term,
s_s is similarity of the time series data,
w_s is the weight of the time series term

Each similarity term gives a similarity measure of 0.0 to 1.0, with a linear combination of weights applied. For example, if three columns are selected, two of which are numeric, and one textual, w_n would be 2/3, and w_t is 1/3. The combined similarity measure remains within a [0.0, 1.0] range.

Numeric Similarity
Similarity between different numeric observations is done by calculating Mahalanobis Distance [15]. This metric is scale-invariant, and ensures that each dimension in the distance gets equal weight. This is very important, as the scales in different numeric fields in a given observation may be quite varied, such as Window Height vs. Total Time vs. Number of Clicks. If distance is calculated without normalizing the data, dimensions with larger numbers would bias the similarity measure.

Textual Similarity
All the tokens extracted from text fields and a standard term vector space model, introduced by [28] for document similarity, is used to determine the level of textual similarity. Each textual field is expressed as a vector of weighted term frequencies and the weights are calculated using a "term frequency-inverse document frequency" model. Similarity is calculated as a cosine between the two vectors.

Time Series Similarity
Time series similarity is done with the widely used Dynamic Time Warping (DTW) algorithm [27] which finds the distance between temporal sequences that vary in time or speed.

The overall strategy in selecting the particular algorithms described above was to allow for fast exploration using different dimensions and slices of the data to find similarity. This approach allowed us to look at just a few of the dimensions, which can reveal patterns that would not be clear if each observation is compared based on all the dimensions.

Some limitations of the current implementation is that we do not have general averaging mechanisms to be able to find similarity between groups of observations. While our numeric and textual similarity approach naturally allows for this to be supported, the time series data is more involved [23]. Also, an important limitation of our time series data similarity is that it is based on the whole signal of a given series. Supporting the ability to find similarity of a selected region of the time series will be a valuable feature in future work.

WORKFLOW EXEMPLAR
To convey the intended usage of UberTagger, we give an exemplary workflow using the data set provided by [3]. The dataset consists of user interaction data such as mouse movement events, clicks, scrolls, etc. The data was extracted from 400 participants filling out a 4 page survey that had two conditions. The survey was designed to test if participants were assisted by visualizations in answering a conditional probability question from the classic decision making problem known as the Mammography Problem [17]. The data set contains both the interaction data and the actual html pages that were used in the experiment. In UberTagger, the

interaction data is used to populate the observation data grid (see bottom left of Figure 1A). Each row represents an answer to one page of a survey, thus for each user there are 4 rows, forming a total of 1600 rows and 20 columns, one for each variable. As the html pages are part of the dataset, as the user selects different rows, the appropriate survey page is loaded into the bottom right panel of the interface.

Such simple, yet useful context visualization linking is achieved in UberTagger by configuration inside of the dataset's package.json, that lets the dataset owner define which observation's data fields would update UberTagger's views. For example, in case of the context webpage, the configuration string is as follows:

```
"context-data":{"url":"data/context/index.html?question={0}&condition={1}",
"args": ["question_id", "condition"]},
```

Similarly, the heatmap visualization (see Figure 1B) is configured to visualize the mouse movements.

General Exploration

An interaction designer opens a dataset and starts exploring it by looking at the observation data grid. She can group it by different variables, such as condition and question ID, and can sort it within these groups. She may look at some time series data of the interactions, scrubbing the timeline, and can see the heatmap change and highlight showing the spatial correspondence of mouse movements with the selected temporal region. Finding an interesting pattern in the time series data, she may tag it simply by entering "#interesting" or "#hmmm" to serve as a bookmark for later investigation.

Tagging the Obvious

To get started she may start by tagging all the answers with "correct" and "exact" tags, to annotate correct responses to the main survey question. Following the workflow of Micallef et al. [17], she may enter a code snippet comment to compute the distance and bias to the exact answer and plot this with a histogram. Having the experimental results, she can now turn to hypothesis generation and develop theories as to why participant performance was so low.

Tagging Textual Comments

The survey also included text input fields that let the participants enter open-ended responses. As the interaction designer starts reading them and tagging them, she may also begin using the auto-tags to save typing and to ensure existing tags will be re-used when possible. She may also select some columns so that similarity searches start finding some existing tags and co-occurrences as well. As the user enters a new tag in the comment panel, co-occurrence search is performed to retrieve any tags that were added before.

Hypothesis Generation

As the analyst goes through the comments, and explores the dataset by looking at the corresponding interaction data, she may formulate some preliminary hypotheses. For example:

- Participants are more confused by the questions then they are by the graph.
- Participants that actually try using the graph, are closer to the correct answer.
- There are similarities between how close the participant answers the main question of the survey, and the follow on questions.

Tagging Other Questions

Not all survey answers had open-ended responses, so once the designer has explored the most complete observations, she can move on to tagging others. As she tags these non-textual observations, tag suggestions speed up her work. The effectiveness of the suggestions is improved by selecting some set of variables so that similarity search finds similar records to extract existing tags and to propose auto-tags.

Similarity Search

To guide the tagging process to be hypothesis driven, when the designer discovers an interesting observation, she can select multiple variables of that observation, and click the "Similarity" button in the selection panel. This updates the similarity column in the grid, where she can sort the observations. This could help to more quickly go through similar observations. This is the same similarity search functionality used to populate the Suggestions panel.

Further Evidence of the Hypothesis

As the designer discovers more evidence for the hypothesis, she adds comments with plots and selections supporting the hypothesis. Also, she starts to add some comments with some statistical summarization and analysis of the data, starting a hypothesis confirmation process.

While this usage scenario is both idealized and specific to the dataset under review, it touches on several common tasks and directions for exploratory data analysis. Future work will include the development of a protocol and controlled datasets to formally evaluate UberTagger.

DISCUSSION

Several frameworks have been proposed to help organize and evaluate visualization, annotation, and analysis systems [7,16,22]. In terms of Heer and Shneiderman's taxonomy of interactive dynamics for visual analysis, UberTagger supports the majority of the "critical tasks that enable iterative visual analysis" [7]. The taxonomy includes twelve tasks grouped into three categories. We have implemented functionality supporting the Data and View Specification category that includes the tasks of Visualize, Filter, Sort, and Derive. In the View Manipulation category, our system supports the Select, Navigate, Coordinate, and Organize tasks. Finally, in the Process and Provenance category, we currently include Record and Annotate tasks. This leaves collaboration functionality for the Share and Guide tasks which, we believe, could also be supported by our general tagging strategy. For example, each user could be represented as a tag category, i.e. a tag parent.

As suggested by the high level of task overlap between UberTagger and Heer and Shneiderman's taxonomy [7], following the ESDA principles [4] was an effective strategy to supply the functionality needed for hypothesis generation. However, when considering the tagging process as the creation of a meta-data set, the analysis of this meta-data would also be beneficial in hypothesis generation. For example, while UberTagger supports the simple meta-tagging of comments because they can be selected, and as they are displayed in chronological order, some meta-analysis may be possible. However, a more sophisticated comment history panel would likely be fruitful to explore, perhaps adding meta-data analysis as a primary task in analytics taxonomies.

While the tasks we performed were enhanced effectively by the embedded recommender system and similarity measures, there are likely more learning and pattern recognition opportunities that could be integrated that would further support hypothesis generation.

CONCLUSION & FUTURE WORK

We have shown how the exploratory analysis and tagging of micro-interaction data can support hypothesis generation. While there are several fundamental data transformations in ESDA, all of the features needed can be supported by adding a rich tagging system (extending a basic system with weights and a parent-child relation) and a rich comment tool (supporting general JavaScript programming). The tag/selection/comment system we have shown can unify an intelligent user interface for a hypothesis generation workflow providing a balance between qualitative and quantitative methods.

We have also shown how suggestions and similar data properties can help label otherwise unlabeled data, promote re-use of tags and avoid subtle tag differences in case or plurality, and form reusable tag structures and hierarchies. If viewed as a lightweight ontology, it may be possible to offer additional suggestions beyond local, similar, and co-occurrence tags, by searching portions of emotion and cognition ontologies [18], for example, for related terms and structures of terms. In broad areas such as medicine, automated hypothesis generation has been considered that also makes use of a mapping to an ontology of medical terms [33].

Automated hypothesis generation, to generate novel and experimentally testable hypotheses, has been developed where a large corpus of scientific literature has been available [32]. However, for the domain of interactive systems, where each issue being examined is likely to be unique, some other expression of design goals may be needed to offer further assistance to interaction designers in hypothesis generation.

REFERENCES

1. Bateman, S., Gutwin, C., Osgood, N., and McCalla, G. Interactive usability instrumentation. *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems - EICS '09*, ACM Press (2009), 45–54.
2. Bostock, M., Ogievetsky, V., and Heer, J. D^3: Data-Driven Documents. *IEEE transactions on visualization and computer graphics 17*, 12 (2011), 2301–9.
3. Breslav, S., Khan, A., and Hornbæk, K. Mimic: visual analytics of online micro-interactions. *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces - AVI '14*, ACM Press (2014), 245–252.
4. Fisher, C. and Sanderson, P. Exploratory sequential data analysis: exploring continuous observational data. *Interactions 3*, 2 (1996), 25–34.
5. Gettys, C.F., Manning, C., Mehle, T., and Fisher, S. Hypothesis Generation: A Final Report of Three Years of Research. *Technical Report 15-10-80*, Decision Processes Laboratory; University of Oklahoma (1980).
6. Guimbretiére, F., Dixon, M., and Hinckley, K. ExperiScope: an analysis tool for interaction data. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*, ACM Press (2007), 1333.
7. Heer, J. and Shneiderman, B. Interactive Dynamics for Visual Analysis. *Communications of the ACM 55*, 4 (2012), 45–54.
8. Heuer, R. *Psychology of Intelligence Analysis*. Center for the Study of Intelligence, 1999.
9. Hilbert, D.M. and Redmiles, D.F. Extracting usability information from user interface events. *ACM Computing Surveys 32*, 4 (2000), 384–421.
10. Kim, J.H., Gunn, D. V., Schuh, E., Phillips, B., Pagulayan, R.J., and Wixon, D. Tracking real-time user experience (TRUE): a comprehensive instrumentation solution for complex systems. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, ACM Press (2008), 443–452.
11. Korotkov, S. TingoDB. 2013. https://github.com/sergeyksv/tingodb.
12. Kort, J. and de Poot, H. Usage analysis: combining logging and qualitative methods. *CHI '05 extended abstracts on Human factors in computing systems - CHI '05*, ACM Press (2005), 2121–2122.
13. Lee, T., Nam, J., Han, D., Kim, S., and In, H.P. Micro interaction metrics for defect prediction. *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering - SIGSOFT/FSE '11*, ACM Press (2011), 311.
14. Loisel, S. Numeric Javascript. 2011. http://www.numericjs.com/.
15. Mahalanobis, P. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta) 2*, 1 (1936), 49–55.

16. Marlow, C., Naaman, M., Boyd, D., and Davis, M. HT06, tagging paper, taxonomy, Flickr, academic article, to read. *Proceedings of the seventeenth conference on Hypertext and hypermedia - HYPERTEXT '06*, ACM Press (2006), 31.

17. Micallef, L., Dragicevic, P., and Fekete, J.-D. Assessing the Effect of Visualizations on Bayesian Reasoning through Crowdsourcing. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2536–2545.

18. Miller, E., Seppa, C., Kittur, A., Sabb, F., and Poldrack, R. The Cognitive Atlas: Employing Interaction Design Processes to Facilitate Collaborative Ontology Creation. *Nature Precedings*, (2010).

19. Miller, G.A. WordNet: a lexical database for English. *Communications of the ACM 38*, 11 (1995), 39–41.

20. Owen, R.N., Baecker, R.M., and Harrison, B. Timelines, a tool for the gathering, coding and analysis of usability data. *Conference companion on Human factors in computing systems - CHI '94*, ACM Press (1994), 7–8.

21. Park, S., Shoemark, P., and Morency, L. Toward crowdsourcing micro-level behavior annotations: the challenges of interface, training, and generalization. *... on Intelligent User Interfaces*, (2014), 37–46.

22. Perer, A. and Shneiderman, B. Systematic Yet Flexible Discovery: Guiding Domain Experts Through Exploratory Data Analysis. *Proceedings of the 13th International Conference on Intelligent User Interfaces*, ACM (2008), 109–118.

23. Petitjean, F., Ketterlin, A., and Gançarski, P. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition 44*, 3 (2011), 678–693.

24. Pohl, M., Wiltner, S., and Miksch, S. Exploring Information Visualization – Describing Different Interaction Patterns. *Proceedings of the 3rd BELIV'10 Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization - BELIV '10*, ACM Press (2010), 16–23.

25. Rigtorp, E. statkit. 2014. https://github.com/rigtorp/statkit.

26. Saffer, D. *Microinteractions: Designing with Details*. O'Reilly, Sebastopol, CA, 2013.

27. Sakoe, H. and Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing 26*, 1 (1978), 43–49.

28. Salton, G., Wong, A., and Yang, C.S. A vector space model for automatic indexing. *Communications of the ACM 18*, 11 (1975), 613–620.

29. Sanderson, P. and Fisher, C. Exploratory Sequential Data Analysis: Foundations. *Human-Computer Interaction 9*, 3 (1994), 251–317.

30. De Santana, V.F. and Baranauskas, M.C.C. Summarizing observational client-side data to reveal web usage patterns. *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*, ACM Press (2010), 1219.

31. Schlueter, I.Z. package.json. 2013. https://www.npmjs.org/doc/files/package.json.html.

32. Spangler, S., Myers, J.N., Stanoi, I., et al. Automated hypothesis generation based on mining scientific literature. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, (2014), 1877–1886.

33. Srinivasan, P. Text mining: Generating hypotheses from MEDLINE. *Journal of the American Society for Information Science and Technology 55*, 5 (2004), 396–413.

34. Stolte, C., Tang, D., and Hanrahan, P. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 52–65.

35. Tukey, J.W. *Exploratory Data Analysis*. Addison-Wesley Publishing Company, 1977.

36. Umbel, C., Ellis, R., and Mull, R. natural. 2011. https://github.com/NaturalNode/natural.

37. Viegas, F.B., Wattenberg, M., van Ham, F., Kriss, J., and McKeon, M. ManyEyes: A Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1121–1128.

38. Wang, R. Node-WebKit. 2013. https://github.com/rogerwang/node-webkit.

39. Weaver, C. Building Highly-Coordinated Visualizations in Improvise. *IEEE Symposium on Information Visualization*, IEEE (2004), 159–166.

40. Willett, W., Heer, J., Hellerstein, J., and Agrawala, M. CommentSpace: Structured Support for Collaborative Visual Analysis. *ACM Human Factors in Computing Systems (CHI)*, (2011).

41. Williams, M. and Norris, T. jstat. 2010. https://github.com/jstat/jstat.

42. Wolf, K., Rohs, M., Naumann, A., Müller, J., Laboratories, D.T., and Berlin, T.U. A Taxonomy of Microinteractions: Defining Microgestures Based on Ergonomic and Scenario-Dependent Requirements. In P. Campos, N. Graham, J. Jorge, N. Nunes, P. Palanque and M. Winckler, eds., *Human-Computer Interaction – INTERACT 2011*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, 559–575.