Learning Custom Experience Ontologies via Embedding-based Feedback Loops

Ali Zaidi[*]
University of Illinois at Urbana-Champaign, Urbana, IL, USA

Kelsey Turbeville[*]
Kristijan Ivančić
Jason Moss
Jenny Gutierrez Villalobos
Aravind Sagar
Huiying Li
Charu Mehra
Sixuan Li
Scott Hutchins
UserTesting, San Francisco, CA, USA

Ranjitha Kumar
University of Illinois at Urbana-Champaign, Urbana, IL, USA
UserTesting, San Francisco, CA, USA

ABSTRACT

Organizations increasingly rely on behavioral analytics tools like Google Analytics to monitor their digital experiences. Making sense of the data these tools capture, however, requires manual event tagging and filtering — often a tedious process. Prior approaches have trained machine learning models to automatically tag interaction data, but draw from fixed digital experience vocabularies which cannot be easily augmented or customized. This paper introduces a novel machine learning interaction pattern that generates customized tag predictions for organizations. The approach employs a general user experience word embedding to bootstrap an initial set of predictions, which can then be refined and customized by users to adapt the underlying vector space, iteratively improving the quality of future predictions. The paper presents a needfinding study that grounds the design choices of the system, and describes a real-world deployment as part of UserTesting.com that demonstrates the efficacy of the approach.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; **Visualization**; *Interaction design.*

KEYWORDS

UX research; usability testing; clickstream analytics; Sankey diagrams; sequence alignment

ACM Reference Format:
Ali Zaidi, Kelsey Turbeville, Kristijan Ivančić, Jason Moss, Jenny Gutierrez Villalobos, Aravind Sagar, Huiying Li, Charu Mehra, Sixuan Li, Scott Hutchins, and Ranjitha Kumar. 2023. Learning Custom Experience Ontologies via Embedding-based Feedback Loops. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), October 29–November 01, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3586183.3606715

[*]Both authors contributed equally to the paper

1 INTRODUCTION

To monitor digital experiences, organizations often rely on behavioral analytics tools [1, 16]. These tools are powerful because they capture most user- and system-level events (e.g., clicks, page loads) that occur on deployed websites and applications. Rendering this data *intelligible* at scale, however — for instance, to measure key performance metrics (KPIs) such as conversion or drop-off rates — typically requires much manual work in the form of event tagging and filtering [8].

To perform this tagging, developers and analysts must instrument the digital asset to map user actions (e.g., click, scroll) to semantic descriptions of user *intents* (e.g., checkout, browse inventory). Although modern analytics tools offer interfaces to assist with this task, it remains a time-consuming process that must be undertaken each time an application — or the analysis to be performed on it — is meaningfully altered. Prior work has demonstrated how machine learning (ML) models can be trained to automatically tag interaction data [17, 25, 28], but these methods draw from fixed digital experience vocabularies which cannot be easily augmented or customized.

This paper introduces a new interaction pattern for event tagging based on ML (Figure 1). The system leverages a general user experience (UX) word embedding to make a set of initial intent predictions for each user action. Then, analysts can correct and refine these predictions through interactive visualizations of the application's analytics data. As more feedback is given to the system, the vector space used to encode the organization's UX ontology is refined and augmented via a counter-fitting algorithm [29], improving the performance of future predictions in turn. In this way, the system allows organizations to employ their own custom semantic vocabularies and extract more relevant insights from user analytics data.

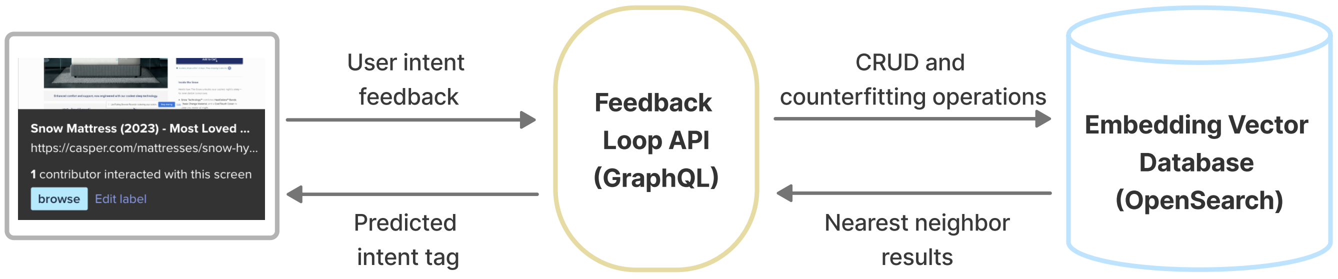
Snow Mattress (2023) - Most Loved ...
https://casper.com/mattresses/snow-hy...
1 contributor interacted with this screen
browse
Edit label
User intent feedback
Predicted intent tag
Feedback Loop API (GraphQL)
CRUD and counterfitting operations
Nearest neighbor results
Embedding Vector Database (OpenSearch)

Figure 1: The feedback loop system comprises three parts: a front-end that captures user feedback; an API that maps the feedback to database and vector space refinement operations; and an embedding vector database that computes nearest neighbor tags — custom and global — for a given element text query.

To inform the design of the proposed system, we conducted a needfinding study with industry professionals who work with web analytics data. The study confirmed the high cost of instrumentation in analytics, and also revealed that tag standardization is another key challenge in large organizations, since different teams may use different terminology to represent similar intents. This observation led to a second, unanticipated use of the system, which can surface related semantic concepts between teams to promote standardization, or allow individual teams to learn their own custom ontologies.

To evaluate the system, we implemented and deployed it as part of UserTesting.com — the industry-leading remote usability testing platform — for six months. Organizations leverage the UserTesting platform to recruit third-party users to perform tasks related to their digital assets. The platform then generates visualizations of the paths users traced, and performs automated tagging of the constituent interactions. We show that the system is capable of predicting custom tags over new interactions from just a few user refinements, and that users most often leverage the relabeling to employ market- or domain-specific concepts. Finally, we illustrate the system's capabilities in two distinct market verticals.

2 RELATED WORK

The goal of this work is not to leverage user feedback to memorize the intent tags of specific UI elements, but to learn semantic representations of these elements that can be used to efficiently categorize the interactions that comprise a digital experience. The proposed feedback loop seeks to mitigate pain points associated with current event tagging tools by leveraging design semantics and interactive machine learning techniques.

2.1 Event Tagging

Organizations primarily rely on external software tools to collect web analytics data and monitor digital experiences. There are many such tools on the market: Google Analytics, Google Tag Manager, Adobe Analytics, WalkMe, ObservePoint, etc. They allow users to manually create and assign tags to digital experiences. These tags can then be used to organize and query specific experiences and compare them. All provide some level of abstraction that helps address common pain points in event instrumentation. Consider Google Tag Manager: while this software does give off-the-shelf capabilities for tracking common events on a digital asset such as when the page has loaded, the workflow becomes exceedingly complicated when custom events become involved. Users must first identify the custom event they want to track, and then create event triggers for that event using CSS element selectors. A user must set up the tags for each event manually, and verify that data is flowing from the Google Tag Manager to the Google Analytics interface. This process must be repeated for each custom event [3].

While each software option provides similar functionality, the overall *look and feel* of one can differ significantly from another [1, 4, 16]. Thus, individuals within organizations consider several smaller factors when deciding which software tool they rely on for extracting design semantics from their user flows. We investigate these factors in detail by conducting a formative interview study with web analytics professionals. The methodology and results of this study are described in Section 3.

Regardless of the specific software used, manual instrumentation is a laborious process. Users must tag each event individually in order for the event to receive the same tag in the future. These systems memorize the tags for the instrumented events and cannot generalize beyond the set of CSS element selectors that are provided. Therefore, the proposed feedback loop utilizes design semantics to tag interactions, allowing the system to automatically recognize classes of UI elements that are semantically similar without manual instrumentation.

2.2 Design Semantics

Prior work has developed approaches for predicting design semantics that describe the *structural* (e.g., map view) and *functional* (e.g., search button, login screen) roles that design elements play in a user experience [13, 26]. These semantics can be applied at all levels of the design hierarchy: UI components, UI screens, and user flows (i.e., screen sequences with interaction transitions).

At the component-level, there are classifiers designed to understand the individual UI/UX elements on a given screen. These classifiers learn what components such as buttons or navigation bars may look like on one screen, and are able to identify these same components on different screens. For example, Liu et al. presented an approach that generated semantic annotations for mobile UI elements [26]. Zhang et al.'s work in mobile UI accessibility extracts and evaluates individual UI elements based on how their design semantics address accessibility heuristics [35].

Rather than predict semantic labels for each individual UI component on a screen, other approaches aggregate component-level features to produce an overall screen-level semantic understanding. Screen2 Words uses human-generated annotations of screens to train a model that produces text-based summaries of an entire mobile UI screen [34]. Each summary encapsulates the purpose of a given screen (e.g. "Page showing different shipping locations" or "Page displaying different languages to choose"). Screen2Vec works at the screen-level granularity as well, generating screen embeddings of mobile interfaces based on textual and graphical elements present on a screen and application-level descriptions [25]. The work demonstrates how these embeddings capture the semantic meaning of UI screens.

Finally, other work has explored predicting design semantics for user flows —- i.e., interaction sequences spanning multiple screens. Wang et al.'s framework utilizes large language models (LLMs) to extract semantic meaning from UI screen interactions [33]. The MoTIF system determines the feasibility of specific user tasks based on input UI screens. For example, MoTIF will determine that "changing the temperature to degrees Celsius" is feasible if screens from a weather application are given, but not if screens from a calculator application are given. This approach requires extracting design semantics from multiple UI screens and bridging them together to determine if an input task is feasible [11].

These automated approaches address some of the issues with manual instrumentation. However, they often rely on fixed vocabularies that cannot be augmented to accommodate custom types of components, screens, or interactions. For example, some of these approaches utilize ontologies that are bootstrapped by the RICO dataset, a large set of mobile application designs [17]. These sets are neither comprehensive nor specialized to a domain, meaning that users or groups with domain-specific ontologies cannot see their vocabularies reflected in these systems. Instead, we propose a feedback loop that is bootstrapped with a general vocabulary, similar to previous work, but allows for the automated expansion of this general vocabulary through user feedback. The proposed system predicts interactions using screens and components, and the feedback loop allows it to generate a more dynamic training set so that future interactions can be tagged with custom language.

2.3 Interactive Machine Learning

The proposed system leverages design principles from interactive machine learning. User control in recommender systems is shown to increase user satisfaction with the suggestions a system presents [22, 24]. This control often comes in the form of "feedback" that a user can provide to the system to indicate their level of satisfaction with that system's output. The granularity of this feedback can vary between critiquing a specific prediction the system makes ("You were wrong to predict that I would like this show") to critiquing a general assumption the system made about the user ("You are wrong in thinking I like comedy shows."). Systems also have to balance between enabling users to *explicitly* leave feedback where they can directly tell the system that a prediction or classification is wrong or correct, and *implicitly* leave feedback where the system will consider its output correct if the user interacts with it (e.g. watching a show the system recommends) and incorrect otherwise [21].

Leveraging user feedback is common in interactive machine learning systems [9, 12, 15, 20, 30]. Allowing for feedback often increases system engagement: if users feel like they can have an impact on a system they use, they are more likely to view that system favorably [7, 32] Thus, the proposed system allows for users to provide both explicit and implicit feedback on digital experience tags. Over time, this feedback customizes the system to the user or group's specifications, tagging digital experience data with their custom vocabulary.

3 NEEDFINDING STUDY

In order to understand how organizations set up event tagging and define experience ontologies, we conducted needfinding interviews with five web analytics professionals. From the subsequent qualitative coding of these interviews, we identified three areas of friction that motivated the proposed system's design: organization workflows, event instrumentation, and tag consistency.

3.1 Methodology

We conducted moderated interviews with five participants based in the United States and Canada who reported using, managing, and/or implementing web analytics as part of their professional responsibilities (two males, three females, median age 34). We recruited the participants using UserTesting, a remote experience testing platform that has a proprietary participant panel, and compensated them 30 US dollars for completing a 30-minute interview. The participants worked in a variety of industries — software, insurance, education, and consulting — and had diverse backgrounds in product management, operations management, engineering, marketing, and art/design. We asked participants to describe their current role, how they use web analytics in their work, and what, if any, challenges they encounter related to web analytics (Appendix A). The research team conducted the interviews remotely, which were recorded and transcribed. The research team used the transcripts to conduct open coding and identify themes.

3.2 Workflows and Tools

The interviews revealed a wide variance in organizational approaches to managing web analytics. Some organizations have a single person manage the entire web analytics process, which includes writing code to track events, setting up the tracking environment, planning which events to track, and analyzing web analytics data (P2, P5), while others have dedicated web analytics teams (P1, P3, P4). Some organizations manage web analytics entirely in-house (P1, P3, P5) while others outsource it to third parties (P2, P4). P4 felt that their current organizational structure led to redundant requests for web analytics data: *"the most time consuming thing I'd say is...people asking for specific reports, people wanting to track this one button... we're tracking it but it's going into this overall report... I think the educating the users piece [is the most time consuming]."* Similarly, P2, who was also responsible for setting up and managing event tags, when asked about their organizational and team structure said *"often times they will make requests from us that are not really doable."* P3 felt that their company's workflow also led to confusion, saying "*there is a barrier of communication of really knowing which team*

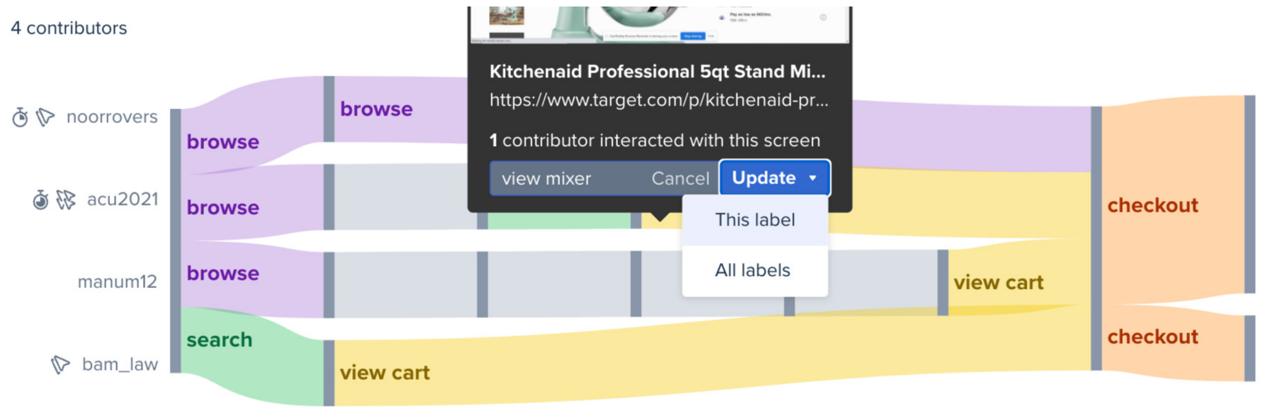

Figure 2: The feedback loop uses a general experience vocabulary to bootstrap initial tag predictions, which are surfaced on an interactive Sankey diagram representing user website navigation paths in the current implementation. Organizations can interact with the diagram to change system-generated intent predictions and add new intents to their experience ontology.

needs what." Both participants who regularly requested web analytics data to consume and participants who fulfilled web analytics requests reported problems related to their team workflows.

Moreover, all participants relied on different web analytics tools and no single software package met all needs. Software mentioned by at least one participant includes: Google Analytics 3/4 ($n = 4$), Contentsquare ($n = 1$), WalkMe ($n = 1$), ObservePoint ($n = 1$), Siteimprove ($n = 1$), Splunk ($n = 1$), and other in-house custom web analytics tools ($n = 2$). Personal preference for specific software was a major driving factor for organizational adoption. In fact, P4 described that their team uses several tools to accommodate individual preferences. They stated that some people within their team *"have allegiance to certain tools and others with other tools... we have like 5 tools that all do the same thing [in our workflow]."*

3.3 Event Instrumentation

Even with extensive use of external software tools for manual instrumentation, many participants (P1, P2, P4, P5) describe instrumenting events, where a user assigns tags to assets or events on a page, as time consuming. For example, P2 described how they instrumented web analytics events to track the number of clicks each news story received on their website: each time a visitor clicked a news story, it would trigger that story's click event, and increment that story's click count. P5 explained that "*setting all the events up [is] very time consuming.*" In addition, P2 described being in a state of *"constant maintenance... we're always looking at... what's useful in our domain... we're always looking for additional features [tags]."*

Finding each instance of an event in code and manually instrumenting it can be a time-consuming activity. To alleviate this pain point, some participants use analytics templates that contain code to recognize different event types and automatically tag and track them. For example, if an organization in the e-commerce sector had many digital assets to track events for, it could create a template for tracking events related to purchasing products ("add to cart", "enter delivery information", etc). Then, each time a new digital asset was created, events related to purchasing would automatically be tagged and tracked. P4 reported that their organization currently used templates which made setting up event tracking much easier: *"it's a lot easier now to deploy new sites because we have sort of a set standard of, you know, events that we want to track."* However, most analytics templates still require customization to detect events that are specific to each organization or market vertical. P4 stated that their current templates are the result of years of manual customization. Some organizations are dissuaded from using templates because the overhead of customizing and maintaining them outweighs the potential benefits. P5 said, *"I would think a template could work, but we do not [use them]."*

3.4 Tag Consistency

With a lack of workflow standardization and many stakeholders, maintaining tag consistency within and across different digital assets can be challenging. P4 stated that "*we had to make sure what we're calling in one site is the same as what we're calling in another site... it could take days to weeks.*" Participants also reported inefficiencies due to inconsistent tracking, and problems arising with new custom tags or changes to existing tags. These changes force organizations to ensure that all corresponding events across websites or products were assigned the new tag: " *we track[ed] internal clicks different than external clicks, but then the directive came down [to] treat them all together... we have to go into the tag manager and delete the internal and external click tag, and create one tag that tracks both... and have to do that on every website"* (P4).

Overall, participants revealed that leveraging current tools still resulted in a lengthy and continuous process of ontology initialization followed by maintenance. Often this process led to tagging inconsistencies within a single organization, which significantly impacted the efficacy of using web analytics. Without tag consistency, participants said that it would be difficult to trust any insights gained from these web analytics.

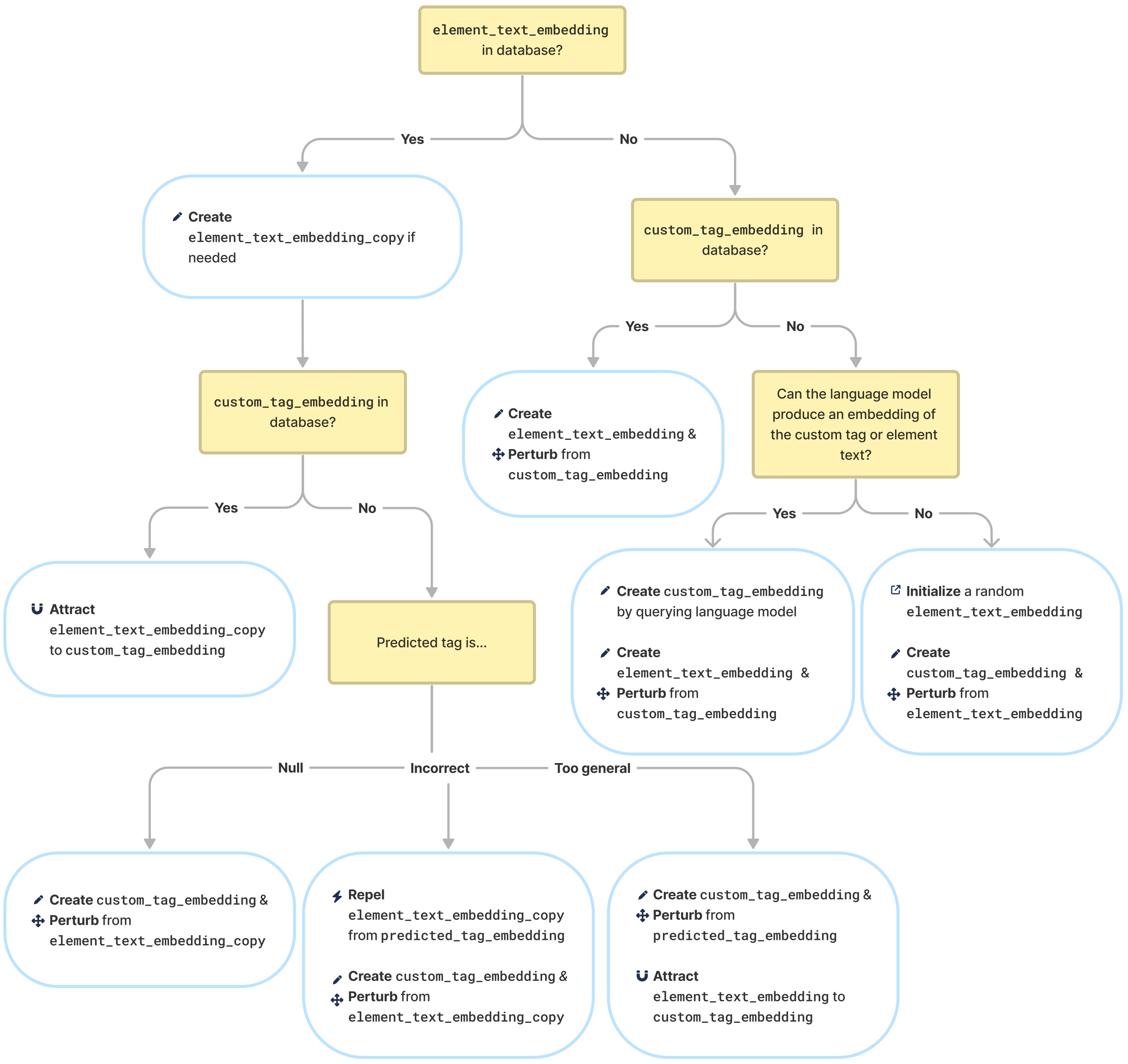
element_text_embedding in database?
Yes
No
Create element_text_embedding_copy if needed
custom_tag_embedding in database?
Yes
No
Create element_text_embedding & Perturb from custom_tag_embedding
Can the language model produce an embedding of the custom tag or element text?
custom_tag_embedding in database?
Yes
No
Attract element_text_embedding_copy to custom_tag_embedding
Predicted tag is...
Create custom_tag_embedding by querying language model
Create element_text_embedding & Perturb from custom_tag_embedding
Initialize a random element_text_embedding
Create custom_tag_embedding & Perturb from element_text_embedding
Null
Incorrect
Too general
Create custom_tag_embedding & Perturb from element_text_embedding_copy
Repel element_text_embedding_copy from predicted_tag_embedding
Create custom_tag_embedding & Perturb from element_text_embedding_copy
Create custom_tag_embedding & Perturb from predicted_tag_embedding
Attract element_text_embedding to custom_tag_embedding

Figure 3: The system leverages vector space refinement operations to add custom tag embeddings to the database and update organization-specific copies of the element text embeddings based on user feedback.

4 CUSTOM UX ONTOLOGY PREDICTION

Based on the needfinding interviews, we designed a machine learning feedback loop for event tagging that facilitates *automation* while also supporting *customization* and *consistency* (Figure 1). This approach relies on an initial set of intent tags and an off-the-shelf language model to bootstrap predictions. Through a feedback interface, users can refine system-generated predictions. The back-end leverages this feedback to add terms to the initial intent ontology and customize the embedding vector representations of its terms. Based on these updates, the back-end can predict semantically similar interactions using the user-defined language. Moreover, we designed this approach to promote consistency when deployed in an organization where many users provide feedback to the system. The system favors used terms more frequently, allowing the organizations to reinforce a cleaner experience ontology over time. We implemented and deployed all parts of the feedback loop as part of UserTesting's platform for remote experience testing.

4.1 Front-end for Capturing Feedback

The UserTesting platform for remote experience testing provides interactive Sankey diagrams to visualize participant journeys through digital assets such as websites. Each bar in the Sankey diagram denotes a UI screen and the edge between bars represents the interaction taken to navigate between the two screens. These diagrams are used in UX research analysis to identify common navigation patterns as well as anomalous behaviors.

We implemented the feedback loop's user interface as part of this feature, overlaying system-generated intent predictions on the Sankey edges, and allowed users to refine them (Figure 2). The system uses an initial set of intent tags and an off-the-shelf language model to bootstrap predictions. For this implementation, predictions are based on the text contained within the elements that a participant interacts with. For example, if a user clicks on a button containing the text "Sign in with Google," that Sankey interaction edge could be tagged with the intent "Sign in." If the element that a participant interacts with contains no text or the system cannot predict an intent for the text contained within it, the Sankey edge will not display any tag.

Users can click on the Sankey edges to update system-generated intents or add an intent where the system did not supply one. User-created intent tags can already exist in the global or custom experience ontology, or may be entirely new. When users update a Sankey edge tagged with a system-predicted tag, they can have that change be applied to all edges with the same system-predicted tag. Additionally, they can provide a rationale for the update: "too general," "inaccurate," or "other." These updates and rationales are then sent to the back-end as feedback for the feedback loop to ingest and modify the embedding space.

4.2 Back-end for Updating Embeddings

The system's back-end modifies the custom embedding space via an API that maps the user feedback to CRUD and vector space refinement operations over an embedding vector database. The vector database comprises embeddings for global tags, which are part of every experience ontology, and organization-specific custom tags. The back-end predicts custom and global intent tags for user interactions as nearest neighbor queries over the vector database.

4.2.1 Global Embedding Space Initialization. We initialize the feedback loop with a "global" experience ontology and an off-the-shelf language model to automatically tag common user interactions. The starting vocabulary comprises 71 common interactions a user might perform when navigating an interface. It includes behavioral and industry-based words and phrases (e.g., "exploration," "collaboration," "purchase," and "favorite").

To ensure that the ontology was comprehensive, we interviewed 10 user behavior analysts about their interaction coding practices. The analysts were recruited from a variety of industries, including travel, e-commerce, and education. From this research, we created a preliminary ontology, which we sent to 15 additional analysts recruited from UserTesting's proprietary panel and asked them to highlight user intents they commonly saw and add any that were missing. Based on this second round of research, we created a final global experience ontology comprising 71 intents, which are part of all custom experience ontologies. If an organization has not defined any custom intent tags, the system will draw all predicted tags from the global ontology.

To bootstrap tag prediction, we insert into the vector database embeddings for each intent in the global ontology and any textual phrases that could appear in interactive UI elements that are often associated with each global intent, which we manually enumerate based on data collected during our research and found in prior work [26]. For instance, "add to cart" is mapped to phrases such as "add to bag" and "add to basket."

We compute embedding vectors for each term in the global ontology based on GloVe [31], a language model used to encode design semantics in prior work [34]. If a global intent tag is one word, we use the pre-trained GloVe embedding vector for this word is used to represent the tag in the vector database. For multi-word tags (e.g. 'Add to cart'), two researchers independently selected a single word from the phrase that best captured its semantic meaning. If both researchers agreed on the representative word, we inserted this word's GloVe embedding into the vector database for the intent tag. If the researchers disagreed, a third researcher broke the tie. This approach is agnostic to the language model we use to compute embedding vectors. For example, in the future, we can use models that accept sequential input to produce embedding vectors such as transformers (e.g., BERT [18]) without modifying the rest of the feedback loop logic.

Once we compute a vector embedding for a global intent tag, we can compute the embedding vectors for its related UI element texts by copying the intent tag's embedding and perturbing it to a nearby point in the embedding space. For example, the embedding for the element text "add to cart" would be near the pre-trained embeddings for the global intent tags "buy" and "purchase." Therefore, when the system performs a nearest neighbor search to predict intent tags, the UI element text queries are mapped to the right intent tags. Incidentally, the global experience ontology and tag prediction architecture could serve as the general template for web analytics workflows mentioned by some participants during the needfinding interviews. Allowing users to start with a set of common tags and tracked interactions could decrease the time spent setting up initial analytics frameworks.

4.2.2 Custom Embedding Space Refinement. While tag prediction over a global intent embedding space obviates initial setup costs, organizations often want to customize event tagging. Therefore, the system allows users to create custom experience ontologies by providing feedback on intent predictions. The feedback loop seamlessly integrates with the tag prediction architecture through an API that maps user feedback to database operations that shape custom embedding spaces for each organization. This approach draws upon prior methods for refining vector spaces based on incorporating new domain-specific terminology and linguistic constraints [19, 29]. While these methods are focused on fine-tuning global vector spaces, we can apply similar techniques at the organization-level to shape each custom embedding space.

Through feedback, users can introduce new custom tags to their experience ontology and redefine the meaning of global and custom tags in the custom embedding space. Since the nearest neighbor search to compute intent for a user interaction is based on the interaction's UI element text embedding, we shape the custom

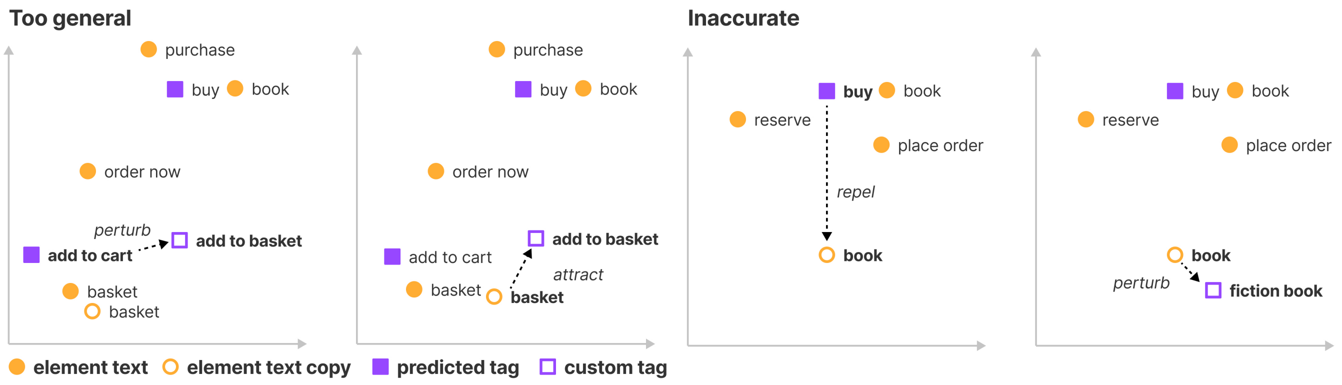

Figure 4: When a user provides feedback that the predicted tag is "too general," the system creates a custom tag embedding *perturbed* from the predicted embedding, and *attracts* the organization-specific copy of the element text embedding to the custom tag embedding. When the predicted tag is deemed "inaccurate," the system *repels* the element text embedding copy from the predicted embedding, and creates a custom tag embedding *perturbed* from the updated element text embedding copy.

embedding spaces—i.e., define the semantic meaning of intent tags—by moving around UI element text vectors. There are four types of vector space refinement operations the API can perform: It can *create* and insert new tag and element text embeddings into the vector database; *perturb* vector representations by adding random noise; *attract* an element text embedding closer to a tag embedding vector to minimize the semantic distance between them; and *repel* an element text embedding away from a tag embedding vector to establish greater semantic distance between them.

The API determines the set of vector space refinement operations to execute based on the user feedback (Figure 3). When a user provides intent feedback, the front-end provides the API with the following data: a user provided `custom tag` describing an interaction, the interaction's UI `element text`, a system-generated `predicted tag` if it existed, and a rationale for the intent update if the user provided one.

If the `element text` embedding already exists in the database:

- The system checks whether an organization-specific copy of the `element text` embedding also exists. If it does not, the system **creates** an `element text` embedding copy and **perturbs** it from the existing `element text` embedding.
- If the `custom tag` embedding exists in the embedding vector database, the system **attracts** the `element text` embedding copy to the `custom tag` embedding to minimize the semantic distance between them.
- If the `custom tag` embedding does not exist in the vector database, the system's actions depend on the value of the `predicted tag`.
  - If the `predicted tag` is `null` (i.e., the Sankey edge initially had no label), the system **creates** a new custom tag embedding and **perturbs** it from the `element text` embedding copy.
  - If the `predicted tag` is `incorrect`, the system **repels** the `element text` embedding copy from the `predicted tag` embedding. The system then **creates** a new custom tag embedding and **perturbs** it from the `element text` embedding copy (Figure 4).
  - If the `predicted tag` is `too general` or no update rationale is provided, the system **creates** a new custom tag embedding and **perturbs** it from the `predicted tag` embedding. The system then **attracts** the `element text` embedding copy to the new custom tag embedding (Figure 4).
  - If the update rationale is `other`, the system takes no action and waits for the update to be reviewed by a person.

If the `element text` embedding does not exist in the database:

- If the `custom tag` embedding already exists in the vector database, the system **creates** a new `element text` embedding and **perturbs** it from the `custom tag` embedding.
- If the `custom tag` embedding does not exist in the embedding vector database, the system's actions depend on whether the language model being used can produce an embedding vector for the `element text` or `custom tag`.
  - If the language model can produce an embedding for the `custom tag` or `element text`, the system **creates** a new `custom tag` embedding by querying the language model. The system then **creates** a new `element text` embedding and **perturbs** it from the `custom tag` embedding.
  - If the language model cannot produce an embedding for either the `custom tag` or `element text`, then the system **creates** a new `element text` embedding and places it at a random point in the embedding space. The system then **repels** the `element text` embedding from this random point, and **creates** a new `custom tag` embedding and **perturbs** it from the `element text` embedding.

4.2.3 Back-end Implementation. Micro-service APIs implemented with GraphQL [5] provide a single interface for vector operations such as inserting/updating/deleting vectors into the database, and finding the nearest neighbors for an element text vector. In the current implementation, when a new custom tag for a previously unseen element text is being added to the vector database, the API will query GloVe for pre-trained word embeddings, which are stored in a separate database. A final custom tag embedding is computed by averaging the GloVe embeddings that are available for all the words contained in the custom tag and its associated UI text.

In order to enable fast vector retrieval, a database powered by OpenSearch [2] stores the embedding vectors. For each embedding vector, the database stores metadata such as the user ID associated with its creation, and whether the vector represents a custom tag or element text. The vectors in the embedding space are all normalized. The perturbation factor in the proposed system is 0.01, meaning that at most a perturbed vector will be 1% different from the original vector. A perturbation factor of 0.01 performed better than 0.1 and 0.001 in our test simulations. The attract function uses cosine distance to continuously pull a vector closer until the distance between the two vectors is below a certain threshold (0.0). The repel function uses cosine distance to continuously push a vector away until the distance between the two vectors is above a certain threshold (1.0).

5 RESULTS

We evaluated the feedback loop's efficacy in two ways: a real-world deployment of the system as part of UserTesting's platform, and case studies that demonstrate how custom intent feedback provided in one test instance can generalize to future tests.

We deployed the machine learning feedback loop in April 2022. Potential feedback loop users could view participant navigation paths as Sankey diagrams with the system tagging interactions as one of the 71 terms that were part of the initial intent ontology. Users could edit and add tags on the Sankey diagram and optionally provide a reason for making the change. Interactions with the system were recorded along with metadata about each user, what custom tags they added, and their reasons for adding these custom tags (if any were provided). Customers of UserTesting were made aware of the feedback loop feature through marketing materials and in-product guides. Additionally, informational articles that taught users how to use the feedback loop feature were added to the UserTesting website [6].

To further evaluate the feedback loop's ability to integrate domain-specific vocabulary into future predictions, we generated case studies for two industries. We recruited participants from UserTesting's participant panel, had them perform navigation tasks on websites relating to these two industries, and recorded their interaction flows in the form of Sankey diagrams. The research team then added custom tags and observed whether intent predictions in future tests run on the same website would surface these custom tags.

5.1 Real-world Deployment

Between April 2022 and December 2022, the system interacted with about 157 million click path events, each represented as a single edge on a Sankey diagram. Of these events, 24 million (15.3%) contained text, meaning that it was eligible to be annotated by the feedback loop. The feedback loop made a total of about 6.8 million intent predictions. Of these predictions, 6.5 million included an intent that belonged to the starting ontology of 71 terms. These statistics reveal the proposed system is able to integrate into the remote usability testing platform's infrastructure and assist users in automatically tagging events using the initial ontology.

Over the course of the deployment, 34 users affiliated with different organizations and companies opted to use this feature. The industries of these 34 users varied. The top five market verticals for this user group were retail (n = 8), transportation (n = 5), healthcare (n = 5), software (n = 4), and IoT/hardware (n = 3). Other verticals included finance, telecommunications, travel, education, real estate, shipping, and consulting.

Users made 546 changes to the initial prediction tags of interactions by adding custom tags. We found that 229 (41.9%) of the custom tags showed up in later predictions: The feedback loop used custom tags when provided. However, users rarely gave reasons for these changes, with only 11 total instances of a user providing feedback. Therefore, we conducted an open coding of instances where users changed a predicted tag from the initial ontology to understand the underlying reasons that users could be providing custom tags.

We examined data from the 10 users (U1-U10) who changed tags most frequently. From these 10 users, we observed 313 instances of customized tags. In 99 of these 313 instances, the system predicted a tag that was a part of the initial ontology, and the user customized it. In 214 instances, the system did not predict a tag. Our codebook consisted of the following justifications: too general, meaning the predicted tag was close to what the user wanted, but not domain-specific enough; incorrect, meaning the predicted tag was completely wrong; and other. Two members of the research team independently coded the set of 99 instances. The inter-rater reliability between the two coders was a Cohen's Kappa of 0.96, indicating strong alignment.

The coding revealed that users often made customizations (n = 64) to change tags to be more specific (i.e., the predicted tag was too general): Typically, the revised tag was a more domain-specific version of the predicted tag. For example, rather than use the predicted tag "search," U5 provided "Find a doc[tor]". Similarly, U8 provided "Product Registration" when the predicted tag was "Create Account".

The remaining instances were either a user correcting an incorrect prediction, or classified as "Other". The system predicted tags incorrectly for two reasons. Sometimes, the system predicted the wrong tag in the initial ontology, such as when U1 felt that a specific interaction should be classified as "browse" rather than "search". Otherwise, the incorrect tag was due to the novelty of the interaction, leading to a poor guess by the system: For instance, the system incorrectly annotated an interaction as "get info" and U2 corrected it to "filter results". All instances coded as "Other" were traced to U6, who switched back and forth between the same two tags within a short period of time, providing "Other" as their reason for making the change each time.

These results demonstrate that users rarely provide explicit feedback, even when they know it will help personalize their experience. In addition, we observe that most of the users customized the Sankey diagrams when predicted tags lacked specificity. Therefore, the proposed system assumes that the predicted tag is too general if users change a predicted tag but do not provide explicit justification feedback. By combining implicit and explicit feedback, we lower the cognitive burden on users while also improving their custom intent embedding spaces over time. This deployment shows that the feedback loop, bootstrapped by the initial ontology, is able to

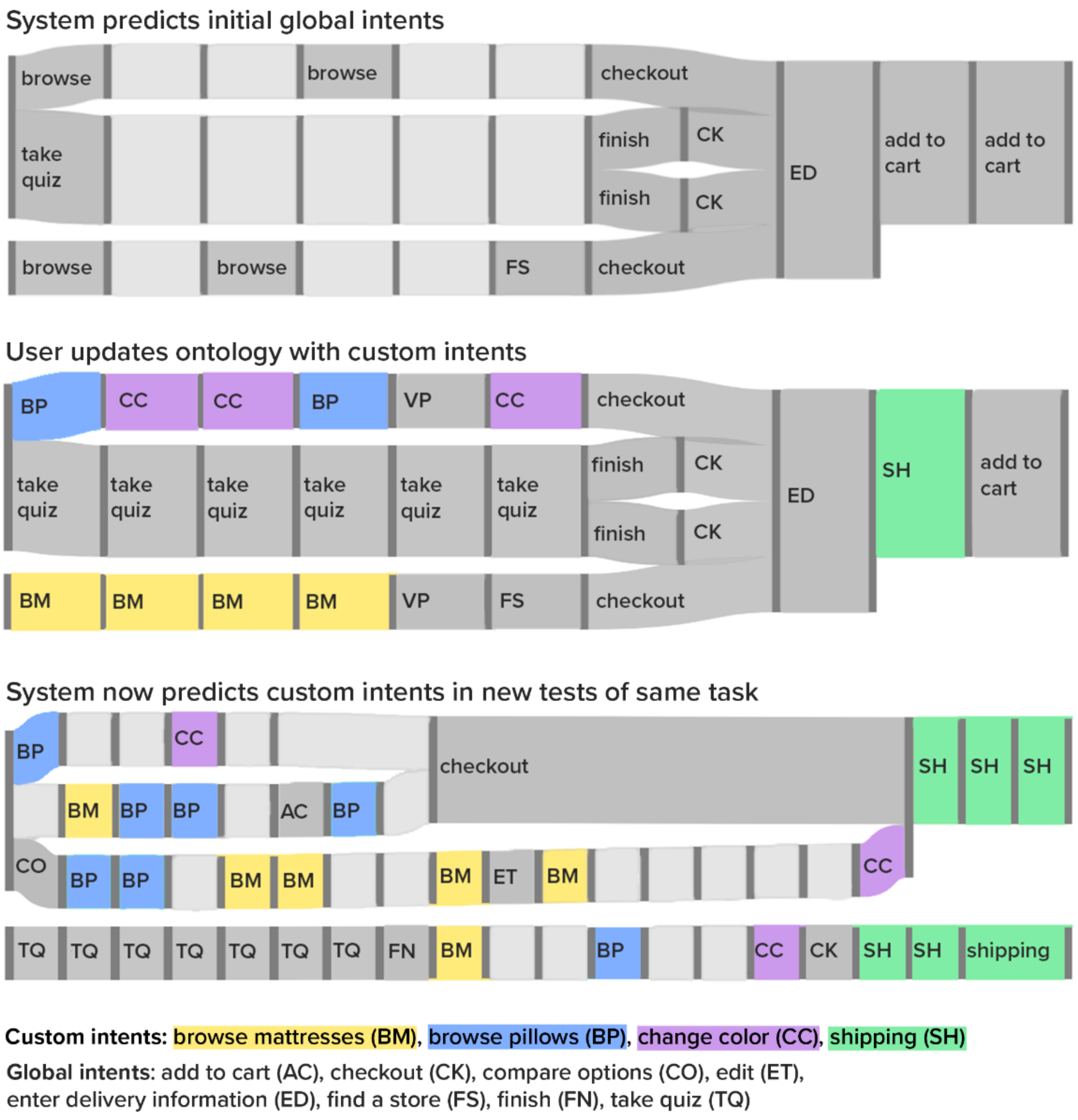
System predicts initial global intents
browse
take quiz
browse
browse
browse
FS
checkout
finish
CK
finish
CK
checkout
ED
add to cart
add to cart
User updates ontology with custom intents
BP
CC
CC
BP
VP
CC
checkout
take quiz
take quiz
take quiz
take quiz
take quiz
take quiz
finish
CK
finish
CK
BM
BM
BM
BM
VP
FS
checkout
ED
SH
add to cart
System now predicts custom intents in new tests of same task
BP
CC
checkout
SH
SH
SH
BM
BP
BP
AC
BP
CO
BP
BP
BM
BM
BM
ET
BM
CC
TQ
TQ
TQ
TQ
TQ
TQ
TQ
FN
BM
BP
CC
CK
SH
SH
shipping
Custom intents: browse mattresses (BM), browse pillows (BP), change color (CC), shipping (SH)
Global intents: add to cart (AC), checkout (CK), compare options (CO), edit (ET), enter delivery information (ED), find a store (FS), finish (FN), take quiz (TQ)

Figure 5: A demonstration of the feedback loop on an e-commerce mattress website usability test where participants were asked to add mattress and bedding to their cart and check out. With just a few user-specified examples of new custom intents, the system can identify semantically similar UI interactions in future tests over the same task.

effectively annotate events on digital assets across organizations in several different industries at a large scale. In addition, when certain events require custom tags, the feedback loop is able to learn from user input and use these custom tags in future predictions.

5.2 Generalization

The real-world deployment demonstrated user interest in using the feedback loop as well as the system's ability to surface custom tags in future predictions. We also demonstrate its end-to-end functionality, starting with initial ontology predictions of events, adding custom tags, and ending with these custom tags surfacing in later tests. We conducted UX tests with digital assets and events in two distinct industries — e-commerce and government — over the course of one week in September 2022. We selected e-commerce and government websites because they express different types of experience intents. For each industry, we selected a website, and asked people recruited from UserTesting's proprietary participant panel to accomplish a website navigation task. The proposed system collected participants' interaction flows through the website interfaces and displayed them as Sankey diagrams with interactions annotated using the initial intent ontology. The research team updated these Sankey diagrams to use a domain-specific vocabulary, along with the reason for each update ("Too general", "Incorrect", or "Other"). We then recruited a new group of participants to ensure their interaction patterns with the website would likely be different and assigned them either the same navigation task or a different one, and observed how the customization persisted across tests: the system was able to leverage the user feedback and annotate user interactions with the custom semantic terms provided in the previous iteration.

5.2.1 Generalizing to the Same Task. The first case study comes from an e-commerce website that sells mattresses and other bedding products. The research team asked four participants recruited from UserTesting's contributor panel to add several items—a mattress, pillow set, and bedding set—to their cart since this is a common user flow for the e-commerce industry.

Initially, the system annotated this flow automatically using the general ontology of 71 terms that bootstrapped the embedding space. Some UI screens where participants viewed products or entered their delivery information were not annotated in the initial Sankey diagram of the user flow. We then manually added custom tags to the Sankey diagram of the user flow, filling in empty parts of the flow and correcting parts that were either too general or incorrect. For example, the initial flow recognized the first quiz screen as a *take quiz* screen, but did not recognize the other pages of the quiz, since the quiz was not a quiz in the traditional sense, but instead tried to provide users with their most compatible mattress. In addition, the research team added domain-specific language such as *browse mattresses*, *change color*, and *browse pillows* to this embedding space.

After adding these custom tags, the research team recruited a new group of four participants to accomplish the same task on that website; participants visited some, but not all, of the same screens to do so. When the Sankey was generated for this new group of participants, it was already filled with the custom tags, demonstrating how the customization can generalize to later tests run by users. After just one round of feedback, the system is able to predict intents using the new custom tags — for interactions with not just the same exact UI elements or elements containing the same exact text —- but for interactions with semantically similar UI elements (Figure 5). This demonstrates that the embedding-based approach does not merely memorize tags but learns the semantic similarity of UI elements.

5.2.2 Generalizing to a Different Task. The second case study is a government website, specifically a United States Department of Motor Vehicles website. On this website, UX researchers might be interested in analyzing how users might find information about getting a learner's permit, or how they might get their first driver's license. We asked a group of five participants recruited from the UserTesting participant panel to find the requirements to get a learner's permit in a particular state. Just as in the first example, the system automatically annotated the user flows from the first test using the general-purpose ontology.The research team then added domain-specific language to the ontology, updating the embedding space to include custom intents such as *find a service*, *permits*, *learning*, and *living*.

To test the feedback loop's ability to understand these new tags, we launched a second test on the same DMV website. We recruited a different group of participants, and instead of getting information about a learner's permit on the website, we asked them to find out how to get their driver's license. This new task would require participants visit different but related screens to the original task.

As expected, some of these custom tags surfaced in the new task's results (Figure 6). We observe that the feedback loop is able to generalize to new tasks, learning UI-specific terminology — *living* and *learning* — and domain-specific tags — *get services*. Given that the task is different, it makes sense that we do not see task-specific tags such as *permits* present in these user flows.

6 DISCUSSION AND FUTURE WORK

Although the proposed system represents a promising step towards a more robust workflow for tag prediction in analytics, there are a number of interesting avenues for future work.

First and foremost among these is improving the predictive power of the embedding at the heart of the system. Although we used GloVe in our implementation, there are several more sophisticated language models that could be employed to more naturally handle multi-word inputs (e.g., BERT [18], GPT-3 [10]). In fact, one of the most fundamental limitations of the presented system is that it makes tag predictions using *only* the text associated with a particular UI element. User interfaces, of course, comprise both visual and textual components: Using a multimodal embedding [14, 23] to learn semantic relationships between words and visual elements in an application could produce more robust predictions. Moreover, since user traces possess temporal context (i.e., the screens that precede and succeed a particular interaction) much like words in sentences have left and right context, it seems plausible that UX-specific embedding algorithms could be developed by generalizing natural language models like Skip-gram or continuous bag-of-words [27].

While our real-world, at-scale deployment did much to validate the utility of the presented approach, a deeper evaluation would aid

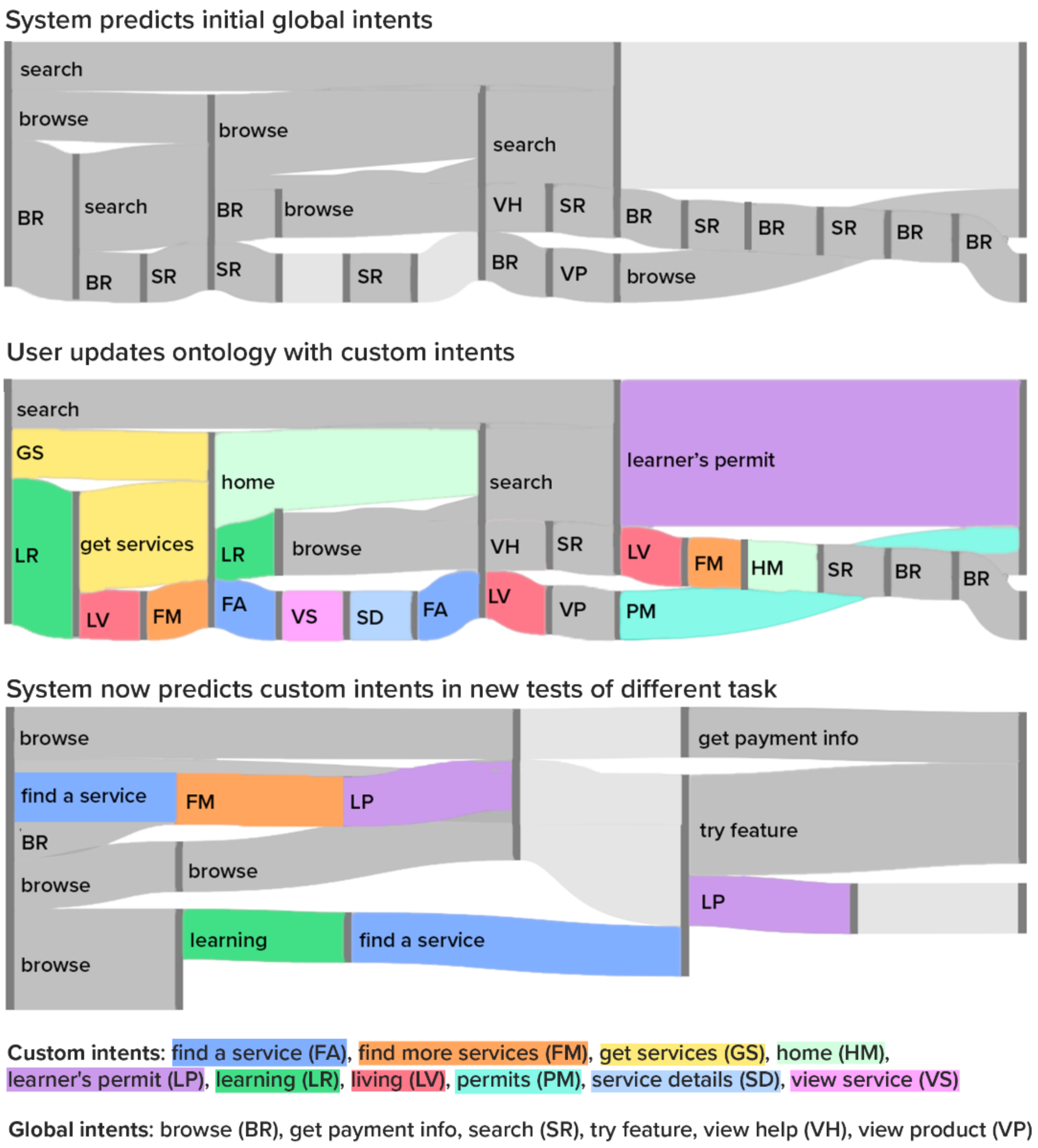

Figure 6: A demonstration of the feedback loop on a Department of Motor Vehicles' website usability test where participants were asked to register for a learner's permit. With just a few user-specified examples of new custom intents, the system can tag semantically similar UI interactions in future tests over different tasks.

future research efforts. Future work should explore additional deployment contexts and navigation activities, as well as solicit more direct feedback from users. For example, a longitudinal study with qualitative feedback from system users would provide significant insight into how the system is used and where it can be improved.

One area of particular interest is better understanding *how* organizations, teams, and individuals develop tagging ontologies in analytics applications. While we designed the proposed system for large organizations, with features grounded in needfinding interviews focused on company-wide experiences, it remains an open question of how much standardization should be encouraged and at what level of organizational granularity. Individuals may benefit from converging on a shared set of core interaction concepts, just as disparate analyses may require different ontological lenses, vocabularies, and terms.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their helpful comments and suggestions. We would like to thank Jerry O. Talton for his generative and discriminative insights; Erin Boehmer and Peter A. Combs for their helpful comments on initial drafts of this paper; and Kaj van de Loo and Akash Sagar for their technical guidance in the early stages of this project. Finally, we would like to thank Jian Jin, Luca Santarelli, Mahsa Forati, Rachad Saab, and Sam Stevens for their efforts in deploying this work in a real system.

REFERENCES

[1] 2022. https://business.adobe.com/products/analytics/adobe-analytics.html
[2] 2022. https://opensearch.org/
[3] 2023. https://tagmanager.google.com/
[4] 2023. https://www.observepoint.com/
[5] 2023. https://graphql.org/
[6] 2023. https://help.usertesting.com/hc/en-us/articles/4403256557076-Intent-Path
[7] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. In *AI Magazine.* 105–120.
[8] Michael Beasley. 2013. *Practical web analytics for user experience: How analytics can help you understand your users.* Newnes.
[9] Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, et al. 2010. Vizwiz: nearly real-time answers to visual questions. In *ACM Symposium on User Interface Software and Technology.* 333–342.
[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *ACM Conference and Workshop on Neural Information Processing Systems.* 1877–1901.
[11] Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. 2022. A Dataset for Interactive Vision-Language Navigation with Unknown Command Feasibility. In *European Conference on Computer Vision.* 312–328.
[12] Joseph Chee Chang, Aniket Kittur, and Nathan Hahn. 2016. Alloy: Clustering with crowds and computation. In *CHI Conference on Human Factors in Computing Systems.* 3180–3191.
[13] Chunyang Chen, Sidong Feng, Zhengyang Liu, Zhenchang Xing, and Shengdong Zhao. 2020. From lost to found: Discover missing ui design semantics through recovering missing tags. In *ACM Conference on Computer Supported Cooperative Work and Social Computing.* 1–22.
[14] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. UNITER: Learning UNiversal Image-TExt Representations. In *European Conference on Computer Vision.* 104–120.
[15] Justin Cheng and Michael S Bernstein. 2015. Flock: Hybrid crowd-machine learning classifiers. In *ACM Conference on Computer Supported Cooperative Work and Social Computing.* 600–611.
[16] Brian Clifton. 2012. *Advanced web metrics with Google Analytics.* John Wiley & Sons.
[17] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In *ACM Symposium on User Interface Software and Technology.* 845–854.
[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 1–16.
[19] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 1606–1615.
[20] Chieh-Yang Huang, Shih-Hong Huang, and Ting-Hao Kenneth Huang. 2020. Heteroglossia: In-situ story ideation with the crowd. In *CHI Conference on Human Factors in Computing Systems.* 1–12.
[21] Dietmar Jannach, Sidra Naveed, and Michael Jugovac. 2017. User control in recommender systems: Overview and interaction challenges. In *International Conference of E-Commerce and Web Technologies.* 21–33.
[22] Yucheng Jin, Bruno Cardoso, and Katrien Verbert. 2017. How do different levels of user control affect cognitive load and acceptance of recommendations?. In *ACM Conference on Recommender Systems.* 35–42.
[23] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. 2014. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. In *ACM Conference and Workshop on Neural Information Processing Systems.* 1–13.
[24] Bart P Knijnenburg, Svetlin Bostandjiev, John O'Donovan, and Alfred Kobsa. 2012. Inspectability and control in social recommenders. In *ACM Conference on Recommender Systems.* 43–50.
[25] Toby Jia-Jun Li, Lindsay Popowski, Tom Mitchell, and Brad A Myers. 2021. Screen2vec: Semantic embedding of gui screens and gui components. In *CHI Conference on Human Factors in Computing Systems.* 1–15.
[26] Thomas F Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. Learning design semantics for mobile apps. In *ACM Symposium on User Interface Software and Technology.* 569–579.
[27] Tomás Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *ACM Conference and Workshop on Neural Information Processing Systems.* 1–9.
[28] Kevin Moran, Carlos Bernal-Cárdenas, Michael Curcio, Richard Bonett, and Denys Poshyvanyk. 2018. Machine learning-based prototyping of graphical user interfaces for mobile apps. In *IEEE Transactions on Software Engineering.* 196–221.
[29] Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting Word Vectors to Linguistic Constraints. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 142–148.
[30] Michael Nebeling, Alexandra To, Anhong Guo, Adrian A de Freitas, Jaime Teevan, Steven P Dow, and Jeffrey P Bigham. 2016. WearWrite: Crowd-assisted writing from smartwatches. In *CHI Conference on Human Factors in Computing Systems.* 3834–3846.
[31] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Conference on Empirical Methods in Natural Language Processing.* 1532–1543.
[32] Kristen Vaccaro, Dylan Huang, Motahhare Eslami, Christian Sandvig, Kevin Hamilton, and Karrie Karahalios. 2018. The illusion of control: Placebo effects of control settings. In *CHI Conference on Human Factors in Computing Systems.* 1–13.
[33] Bryan Wang, Gang Li, and Yang Li. 2023. Enabling conversational interaction with mobile ui using large language models. In *CHI Conference on Human Factors in Computing Systems.* 1–17.
[34] Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. 2021. Screen2words: Automatic mobile UI summarization with multimodal learning. In *ACM Symposium on User Interface Software and Technology.* 498–510.
[35] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, et al. 2021. Screen recognition: Creating accessibility metadata for mobile applications from pixels. In *CHI Conference on Human Factors in Computing Systems.* 1–15.

A NEEDFINDING INTERVIEW SCRIPT

A.1 Introduction

Thank you for participating in this study. My name is [NAME], and I'm a researcher doing a study on how people use web analytics. I'm doing this study to understand how web analytics are used in other companies and what problems currently exist.

I'm going to ask you a series of questions regarding your experiences with web analytics instrumentation, and what problems you feel there are. We are not testing your knowledge in any way.

Please remember to speak your thoughts out loud and be honest with your feedback, both positive and negative.

There are some team members listening in on the session as well, and we're all looking for your feedback to help us improve. They are here to take notes but this session will be mostly a conversation between you and me. As a reminder, we are recording the session so that we can go back later and make notes, but please know that this recording will only be accessible to people working directly on this project. Is that ok with you? [*If the participant declines, terminate the interview.*]

A.2 Background Questions

(1) What is your current role? What are your day-to-day responsibilities?
(2) Are web analytics a part of your work?
  (a) If so, how?
(3) How long have you been working with web analytics?
(4) Describe your general workflow with web analytics.
(5) What are some of the web analytics tools you use?
  (a) If more than one tool: What would you say is your most preferred tool/software and why?
(6) Have you ever been involved in setting up a web analytics platform for your work? Please describe the process.

A.3 Specific Web Analytics Experiences

(1) How long did it take for you to set up the web analytics platform for your work, from downloading the software to having it fully customized?
(2) Now, thinking about your experience in this field more broadly, what, if anything, is the most frustrating/time-consuming part of instrumenting web analytics?
(3) Did you have problems getting the web analytics framework set up? If so, what were these problems?
(4) If possible, can you describe some of the specific events you want your web analytics to track?
(5) How important is the terminology/tags to the analysis process? What would happen if you did not define the terms/tags?
(6) How often do you add new terminology to your web analytics?
  (a) How long does it take for them to be used correctly?
  (b) Who, if anyone, defines the terms?
  (c) Do different people on your team ever use different terms or tags for the same thing when working with your web analytics software? If so, what is the impact of that?