Affinder: Expressing Concepts of Situations that Afford Activities using Context-Detectors

Ryan Louie
Northwestern University
Evanston, IL, USA
ryanlouie@u.northwestern.edu

Darren Gergle
Northwestern University
Evanston, IL, USA
dgergle@northwestern.edu

Haoqi Zhang
Northwestern University
Evanston, IL, USA
hq@northwestern.edu

ABSTRACT

Context-aware applications have the potential to act opportunistically to facilitate human experiences and activities, from reminding us of places to perform personal activities, to identifying coincidental moments to engage in digitally-mediated shared experiences. However, despite the availability of context-detectors and programming frameworks for defining how such applications should trigger, designers lack support for expressing their human concepts of a situation and the experiences and activities they afford (e.g., situations to toss a frisbee) when context-features are made available at the level of locations (e.g., parks). This paper introduces Affinder, a block-based programming environment that supports constructing *concept expressions* that effectively translate their conceptions of a situation into a machine representation using available context features. During pilot testing, we discovered three bridging challenges that arise when expressing situations that cannot be encoded directly by a single context-feature. To overcome these bridging challenges, Affinder provides designers (1) an *unlimited vocabulary search* for discovering features they may have forgotten; (2) *prompts for reflecting and expanding* their concepts of a situation and ideas for foraging for context-features; and (3) *simulation and repair tools* for identifying and resolving issues with the precision of concept expressions on real use-cases. In a comparison study, we found that Affinder's core functions helped designers stretch their concepts of how to express a situation, find relevant context-features matching their concepts, and recognize when the concept expression operated differently than intended on real-world cases. These results show that Affinder and tools that support bridging can improve a designer's ability to express their concepts of a human situation into detectable machine representations—thus pushing the boundaries of how computing systems support our activities in the world.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; *Interactive systems and tools*; *User studies.*

KEYWORDS

context-aware programming; expressing concepts of situations; context-features; design fixation; bridging challenges; block-based programming

ACM Reference Format:
Ryan Louie, Darren Gergle, and Haoqi Zhang. 2022. Affinder: Expressing Concepts of Situations that Afford Activities using Context-Detectors. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3491102.3501902

1 INTRODUCTION

Context-aware technologies have an enormous potential to be helpful and responsive within many situations that arise during people's daily lives. Increasingly, mobile context-awareness applications are being developed to help end-users think about places they are visiting, and what they can do there. For example, such applications can remind users to engage in personal activities or routines (e.g., buy vegetables, listen to live music) based on relevant places they encounter in their daily lives [8] or help users find coincidental moments to engage in shared experiences with other people across distributed contexts (e.g., when family members living apart can share a meal together) [32] The proliferation of context-aware applications have been possible due to the advances in better mobile sensors, location-based information sources (e.g., Foursquare, Yelp), and machine learning algorithms—which have made available a diverse set of component detectors that infer semantically-meaningful aspects of a user's context, which we refer to as *context-features* (e.g., whether a user is moving or stationary, whether they are visiting a park, whether their current weather is windy or not). By providing such semantically-meaningful context-features to program with, frameworks for building context-aware applications have made it easier for application designers to define how an application should trigger and act based on a user's current context.

Despite this focus on better component detectors and context-features, it is difficult to encode human concepts of a situation into a machine representation using available context-features. While context-features provided by mobile context frameworks are useful for detecting events or actions at the level of locations and place categories (e.g., a restaurant tagged with the place category 'soup'; a 'park'), its difficult to use such context-features to describe situations that would support experiences (e.g., 'enjoying a warm meal on a cold day'; 'good for tossing a frisbee') when these concepts are several levels of abstraction removed from the context-features. While context-programming frameworks and trigger-action programming tools [13, 40] have made it easier for authors to access a variety of context-features, their processes for programming the situational detectors are limited to creating simple situation detectors at the level of the events and locations which can be detected. Instead, we argue that what is needed are programming environments that explicitly support the cognitive work required to flesh out an author's concept for how a situation might enable experiences (e.g., 'soup' is one category of restaurants for a cold day, but where else?) and translate how available context-features apply to their concept. If designers could encode their human concepts of a situation—such as what contexts are appropriate for engaging in a personal activity, or what contexts support engaging in a digitally-mediated shared experience—it would improve the ability of applications to recognize human situations and facilitate appropriate activities within them.

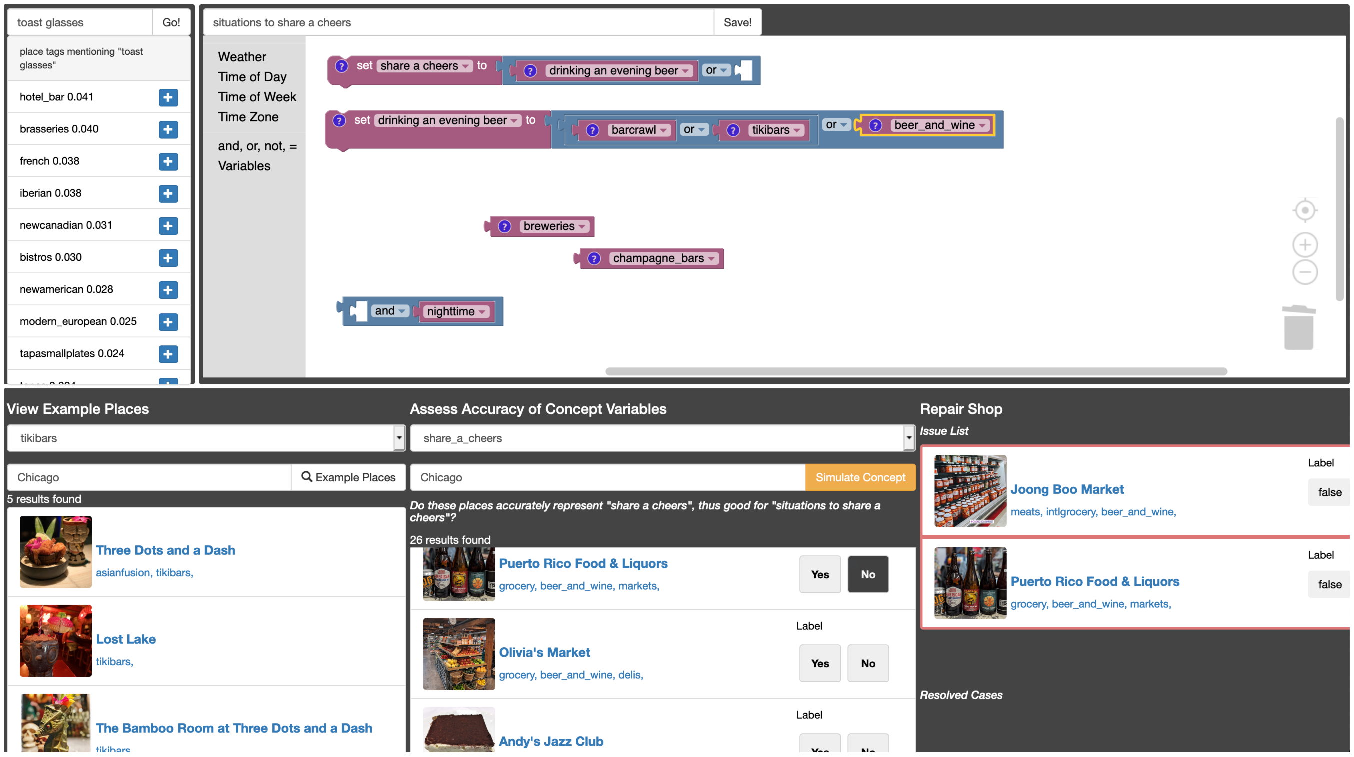
To support designer's in expressing the human concepts of how situations afford activities and experiences to machines, we designed and built a programming environment named Affinder. Using Affinder, a designer can take their ideas for a situation they want to use in a location-based, context-aware application, and translate their human concepts of that situation into a logical expression using readily detectable context-features, which we call a *concept expression*. Concept expressions can then be used by applications to identify the situation across end-users' mobile contexts. For example, a designer of a mobile app that reminds users of opportunities to engage in personal activities or routines can construct the high-level concept "awesome for tossing a frisbee around" by using simpler concepts and detectors such as open recreational areas (disc golf, parks, playgrounds, beaches), weather is not disruptive (not windy), and while there is daylight (time between sunrise and sunset). Affinder was designed to structure a construction process that allows people to flexibly switch between breaking down a higher-level concept into a construction that gets closer to the detectable features (top-down), and in defining more general concepts that link detectable context-features to human concepts (bottom-up). To support this, Affinder uses a block-based programming environment that provides a single, visual workspace for authors to (1) declare concept variables, or intermediate concepts that serve as links between an abstract concept and the context features; (2) forage for context-features that match their concepts; and (3) compose representations using logical operators; see Figure 1.

From our design-based research process creating Affinder, we also uncovered a set of **bridging challenges**, or specific obstacles that arise when trying to express concepts of a situation that cannot be directly specified with any single, detectable context-feature. First, concept expressions may be too narrowly defined, or *underscoped*, when a designer retrieves one context feature for a concept (e.g., 'parks' for 'grassy fields') but misses other context features (e.g., 'football' fields) that also match the concept. Second, the underscoping problem can also occur if a designer fixates on their early concepts of a situation (e.g., 'grassy fields' are good for frisbee tossing), which may result in a too narrow of an effort to forage for context-features that are relevant to the situation (e.g., in narrowly searching for context-features matching 'grassy fields', a designer can miss 'beaches' that are also good for frisbee tossing). Third, concept expressions may also include *detector inaccuracies* when a designer uses a context feature that does not evaluate the concept expression as expected (e.g., 'parks' may match all parks, including dog parks and skate parks that may be less desired as places for tossing a frisbee). To address these bridging challenges, Affinder contributes three core features that support designer cognition when expanding their concepts and translating them to machine features: (1) an *unlimited vocabulary search* for discovering context features they may have forgotten; (2) *reflect and expand prompts* that help designers generalize their notions of the concept they are trying to express and expand their efforts to forage for the context-features; and (3) *simulation and repair tools* for identifying and resolving issues with how machine detectors may operate on real use-cases differently than an author intends.

We conducted a between-subjects test comparing authors' use of Affinder with all of its core features to address the bridging challenges vs. a baseline version of the block-based construction environment that only supported an opportunistic construction process. Across both versions, we found that all participants could use Affinder to encode a conceptual-rich situation (e.g., awesome for tossing a frisbee; enjoying food for a cold day) into a detectable machine representation using available context-features. Participants using the experimental condition of Affinder were able to bridge to several relevant context-features matching their original conception using the unlimited vocabulary search. Additionally, participants expanded their concepts for how a situation could be realized; this conceptual stretch occurred both (a) after discovering context-features during searching that shifted their original notions, and (b) after stopping when stuck and fixated, and then reflecting about how a context-feature is generally appropriate and matches the high-level situation. Finally, the simulation and repair tools helped them recognize and address several cases where their concept expressions and underlying machine detectors operated differently on real-world cases than they expected.

In summary, this paper makes the following contributions:

- We present Affinder, a programming environment that supports an effective process for encoding human concepts of situations into a machine representation using available context features.
- We identify three bridging challenges arising from the mismatch between human concepts of a situation and the machine's available representations, including (1) underscoping on machine features; (2) underscoping on author concepts; and (3) concept expression inaccuracies.
- To address the bridging challenges, we contribute the design of three core features in Affinder: (1) unlimited vocabulary search for discovering relevant machine representations; (2) prompts for reflecting and expanding conceptual representations; and (3) tools for simulating and repairing inaccuracies in how concept expressions operate in real-world cases.
- In a lab study with Affinder, our results show how participants can follow an effective process for constructing concept expressions, and how Affinder's core features can address the bridging challenges by helping authors reflect on, stretch, and update their own conceptions while accurately translating their conceptual representations to the machine's available representations.


toast glasses Go!

place tags mentioning "toast glasses"

hotel_bar 0.041
brasseries 0.040
french 0.038
iberian 0.038
newcanadian 0.031
bistros 0.030
newamerican 0.028
modern_european 0.025
tapasmallplates 0.024

situations to share a cheers Save!

Weather
Time of Day
Time of Week
Time Zone
and, or, not, =
Variables

set share a cheers to drinking an evening beer or

set drinking an evening beer to barcrawl or tikibars or beer_and_wine

breweries

champagne_bars

and nighttime

View Example Places

tikibars

Chicago Example Places

5 results found

Three Dots and a Dash
asianfusion, tikibars,

Lost Lake
tikibars,

The Bamboo Room at Three Dots and a Dash
tikibars

Assess Accuracy of Concept Variables

share_a_cheers

Chicago Simulate Concept

Do these places accurately represent "share a cheers", thus good for "situations to share a cheers"?

26 results found

Puerto Rico Food & Liquors
grocery, beer_and_wine, markets,
Yes No

Label
Olivia's Market
grocery, beer_and_wine, delis,
Yes No

Label
Andy's Jazz Club

Repair Shop

Issue List

Label
Joong Boo Market
meats, intlgrocery, beer_and_wine,
false

Label
Puerto Rico Food & Liquors
grocery, beer_and_wine, markets,
false

Resolved Cases

Figure 1: Affinder is a block-based programming environment for constructing concept expressions that effectively express a designer's concepts of a situation and the activities it affords (e.g., situations to share a cheers) to machines using available context-features. The visual workspace (top-right) supports declaring intermediate concepts that serve as links between an abstract concept and available context features; foraging for context features through browsing and searching hierarchies of features; and composing representations. Affinder implements three core features to support construction: (1) an *unlimited vocabulary search* tool (top-left) helps designers discover available context-features relevant to their concepts; (2) *reflect and expand prompts* help designers generalize their notions of the concept they are trying to express and expand their foraging efforts; and (3) *simulation and repair tools* (bottom) help with identifying and resolving issues with the precision of concept expressions on real use-cases.

In the rest of the paper, we motivate Affinder through related work, describe the bridging challenges that arise during construction, and we detail the design and implementation of Affinder. We then report on the results of the lab study of Affinder, and end with a discussion on takeaways and future directions for tools that support bridging between human concepts and machine representations.

2 BACKGROUND

Context-aware systems are made up of two main components: (1) context providers, which use algorithms and inference techniques to extract attributes of a persons' context from sensors, which we refer to as machine detectable context-features; and (2) context-awareness services, which continuously reason about these attributes of context to perform useful actions on behalf of the user [36]. To develop use-cases for context-aware applications, authors must encode their concepts of a situation into a machine representation using context features made available by context providers. Since the created machine representations are composed of detectable context-features, context-aware services can then use these machine representations to act and facilitate interactions within desired situations.

Over the last two decades, research in context-aware computing and machine learning has significantly expanded the ability of context providers to infer aspects of human context across the dimensions of location, identity, activity, and time [1]. This work has led to numerous component detectors for various facets of context through better data, algorithms, and sensors (e.g., [22]). More recently, research systems have focused on using machine learning to model complex human situations within the domains of human activity recognition [18, 27], interruptibility and optimal work breaks [26], and mood-related mental health [34].

Today, many context features are widely available through mobile context providers (e.g., the Google Awareness API [29]) that implement a wide range of component detectors including time, location, places, activity, and weather. Moreover, context-aware and trigger-action programming tools (e.g., iCAP [13], and IFTTT, or if-this-then-that [40]) have made it easier to program with context features. But while existing context provider APIs and programming tools provide application designers access to large sets of context-features, they provide few supports for expressing and encoding higher-level concepts of a situation that may be several levels of abstraction removed from these features. For example, while several trigger-action frameworks (e.g., multi-trigger variants of IFTTT) do support composing multiple context-features together into a single event, the situations expressed are assumed to be near the feature level, which allows rules to be built by accessing conceptual features directly [40].

When concepts of a situation are difficult for users to encode with the available features, programming by demonstration approaches have been successful in helping users map between a high-level situation and the available context features and sensors [11]. However, these approaches assume human teachers have an unchanged understanding about a concept and thus can easily provide positive and negative examples to a model to support. For our setting and task of expressing an author's concept of situations that support a desired activity, the application designer can suffer from design fixation [3, 7], which could result in the created machine representation being underscoped, or too narrowly defined, with respect to all the ways a situation could support a desired activity. Ultimately, unless we help the human expand their conception, whatever way they express it to machines (explicitly through construction, or implicitly through labels) will be limited to their current conception. For our proposed solution, Affinder provides specific cognitive scaffolds for the construction process, where an author declares their concepts and forages for context-features explicitly, which we argue naturally helps them flesh out and expand their own ideas in the process.

One particular challenge that arises in the construction process is identifying context features that support engaging in an experience or activity. Prior work by Dearman et al [9] attempted to do this by identifying potential activities supported at various locations by mining community-authored content (e.g., reviews). This approach extracts verb-noun pairs from community-authored content about a location in order to identify the potential activities supported by that location. While this work creates an extensive set of component detectors for potential activities supported by a location, it is limited to exact activities (e.g., drink soup) but can struggle to return results for higher-level situations of interest (e.g., places to enjoy warm food on a cold day). In other words, their approach may be particularly useful for finding context features matching specific (low-level) activities, but less useful for identifying features related to higher-level situations of interest. To address this problem, we introduce a more flexible unlimited vocabulary approach for finding features, and additionally introduce tools for representing and acting on relevant concepts across levels.

3 THE BRIDGING PROBLEM

In this section, we introduce the idea of the bridging problem: the difficulty in encoding human concepts of how a situation supports desired experiences into a machine representation using available context-features. For example, an author may want to design a situational trigger that identifies everyday opportunities for users to perform a playful activity like throwing a frisbee. Starting with their human concepts, an author wants to include other place contexts like 'parks' good for tossing a frisbee, and also recognizes that most 'open fields ' would be generally appropriate for this activity. Now, the author needs to figure out how to link between their concepts of how a situation enables tossing a frisbee and the detectable context-features.

A key technical challenge is figuring out an effective *construction* process for bridging from human (mental) representations of a conceptually-rich human situation to a machine representation built using available context features. In one direction, a top-down process can support a creator decomposing a situation into simpler concepts, but can frequently lead to scenarios where a creator finds that there are no matching context features for such concepts (e.g., no detector for 'open space'). In earlier systems such as iCAP [13], such concepts are simply ignored and left out of the construction. In another direction, bottom-up approaches allow for reusing and composing context features [37, 40], but developers can become stuck when they do not know a priori what context features may be useful for expressing an abstract idea that they have. To overcome these challenges, we recognized that authors need a more fluid construction process that allows them to choose a top-down or bottom-up strategy in order to find a link between their concepts of the situation and the available context-features.

In addition to enabling a more fluid construction process, we focused our design efforts on uncovering and addressing some of the specific challenges that arise in this construction process. We used a design-based research method in which we iteratively prototyped and tested with participants across two rounds of pilot tests (N = 7, N = 6) to understand how Affinder supports an author's process in constructing high-level situations from available context-features, and any remaining obstacles that arose. We tasked participants with expressing high-level situations that identify opportunities to have shared experiences in a context-aware social application use-case (e.g., situations to share a cheers; awesome situations to throw a paper airplane; situations to watch the sunset over water).

Through this phase of iterative prototyping and pilot testing, we identified three general bridging challenges authors face when there is a mismatch between their human (conceptual) representations and the machine's available representations. Across these challenges, we found that they can be caused by cognition difficulties in fleshing out concepts for how a situation supports potential activities, and in translating these concepts into available machine detectable features.

3.1 Underscoped on Features

Concept expression may be too narrowly defined, or *underscoped*, when a designer recalls one context feature for a concept ('parks' for open space) but forgets other context features (e.g., 'beaches') that also match the concept. The underscoping problem can occur when trying to translate from their concepts to the available features: when foraging for relevant context-features from a long list of features, designers may fail to find a comprehensive set of context-features matching their concept for a situation; see top-row of Figure 2.

We noticed in early piloting that the list of place category detectors provided by the Yelp Places API [24] was extensive and

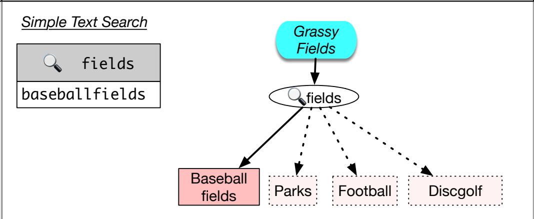
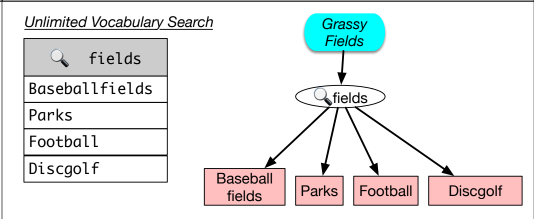
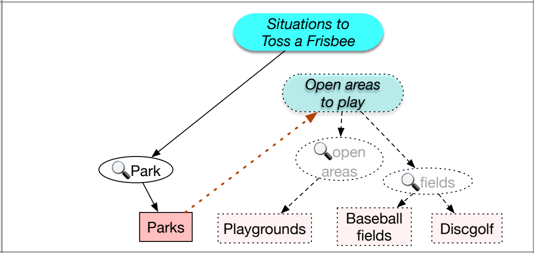
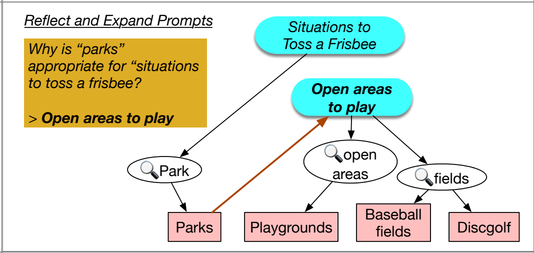
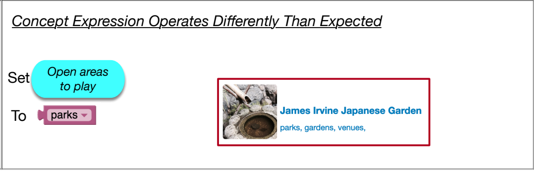
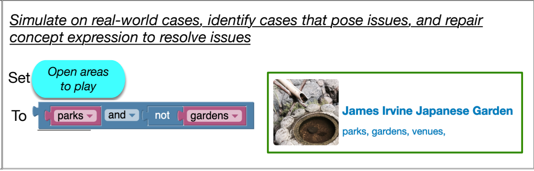
Construction Challenge	*Example of Construction Challenge*	*Affinder's Technical Solution*
Underscoped on Features Designer retrieves some context-features, but misses others that also match the concept	 Simple Text Search fields baseballfields Grassy Fields fields Baseball fields Parks Football Discgolf	 Unlimited Vocabulary Search fields Baseballfields Parks Football Discgolf Grassy Fields fields Baseball fields Parks Football Discgolf
Underscoped on Concepts Designer has narrow notions of the concept they are trying to express, resulting in a limited effort to forage for relevant context-features.	 Situations to Toss a Frisbee Open areas to play Park open areas fields Parks Playgrounds Baseball fields Discgolf	 Reflect and Expand Prompts Why is "parks" appropriate for "situations to toss a frisbee? > Open areas to play Situations to Toss a Frisbee Open areas to play Park open areas fields Parks Playgrounds Baseball fields Discgolf
Concept Expression Inaccuracies Designer may not realize that context-features will operate differently than they intended.	 Concept Expression Operates Differently Than Expected Set Open areas to play To parks James Irvine Japanese Garden parks, gardens, venues,	 Simulate on real-world cases, identify cases that pose issues, and repair concept expression to resolve issues Set Open areas to play To parks and not gardens James Irvine Japanese Garden parks, gardens, venues,

Figure 2: We identify three construction challenges, and implement their corresponding solutions in Affinder: (A) concept expressions can be underscoped if designers fail to retrieve relevant features for a concept; (B) concept expressions can be underscoped if *concept variables* are too narrowly defined which limits efforts to forage for features; and (C) concept expressions can have inaccuracies, when it executes not as a designer intends.

thousands of items long, making it impractical for designers to comprehensively scan through the hierarchy of items. Designers could use simple text search on this list, but they may not know all the names of the place categories that Yelp defines a priori to know which ones will be useful. For example, a pilot participant thought that "fields" would be a type of place listed on Yelp, and with simple text search, this query returned "baseballfields". However, there are many other place categories (e.g., parks, football, stadiums, discgolf) which may contain the open grassy field for frisbee tossing they conceptualized. Evidently, different designers will have their own notions and vocabulary of a concept which makes it difficult to find relevant machine features [17]. Thus, our first round of iterative development and piloting aimed to support designers in finding the broader set of place context-features based on their notions of a high-level activity and situation.

3.2 Underscoped on Concepts

The underscoping problem can also occur when designers fixate on a narrow set of intermediate concepts to describe their overall concept for a situation. Studies of designer's cognition and metacognition highlight that designers have a tendency to fixate their search for solutions, with a cognitive bias towards their earliest solution ideas [3, 7]. Much like the literature suggests, we saw this with participant designers in our pilots. For instance, the middle-row of Figure 2 illustrates how one designer from our pilots narrowly conceptualized the idea of "situations to throw a frisbee". They started by recalling "parks" as a common place they associate with being able to perform the activity. However, after using this search term to find a few place context-features that afforded the activity (e.g., park, dogpark), the participant stopped their search for other potentially relevant place category context features.

What this participant didn't conceptualize was a more generalized notion of what makes a place good for tossing a frisbee (e.g., a place must have "open areas to play"). Had they had a broader notion, they could have continued to forage for context-features in other part of conceptual space (e.g., "open area", "fields"), and found other context-features relevant to detecting situations to toss a frisbee (e.g., playgrounds, baseball fields, discgolf). Thus, our second phase of pilot development focused on helping designers push past their earliest concepts for a high-level situation, and encourage them to expand their concepts and associated context-features for a situation.

3.3 Concept Expression Inaccuracies

Creating a concept expression that operates accurately in real-world scenarios requires iterative refinement. Concept expressions can operate inaccurately when a context-feature does not operate as a designer intends. For example, they may not precisely match a concept (e.g., public gardens are parks but are not good for frisbee tossing); see bottom-row of Figure 2.

In early prototyping, we observed that users' mental-model about a context-feature, based upon only its name, can often mismatch how a context-feature actually applies to real-world cases. This led to participants adding context-features which inaccurately matched the concept expression when applied to real-world place venues (e.g., 'recreation' refers to indoor recreation centers and gyms which are not appropriate for frisbee tossings). As a low-fidelity solution for this challenge, we allowed authors to reference the Yelp website to view example locations that were listed for any place category they were uncertain about. Eventually, we incorporated this as an integrated interface in Affinder, where users could view several example location venues listed for a place category context-feature.

Additionally, a more subtle way concept expressions can inaccurately match was when several real-world cases challenged a designer's primary mental model of a context-feature. For example, a user was originally thinking of "parks" as a category for throwing paper airplanes, but recognized through foraging for other place category context-features that some "parks", such as "dog parks" would more likely lead to "a dog chewing up the airplane." Thus in our second round of pilot prototyping, we focused on developing tools integrated within Affinder's construction environment to help designers simulate their concept expressions on real-world cases so as to surface when concept expressions and their context-features operate differently than intended.

4 AFFINDER

In this section, we introduce Affinder, a programming environment for constructing *concept expressions* that effectively translates an author's human concept of a situation into a machine representation using context-features that can be acted upon computationally. Specifically, Affinder allows a designer to take a situation they wish to use in a location-based or context-aware application (e.g., where to go to share a warm meal on a cold day), and to translate it— through the process of construction—into a logical expression based on available context features that can be readily detected and used with an application (e.g., restaurants serving soup OR restaurants serving spicy food).

To support this, Affinder provides (1) a *block-based programming environment* that facilitates an opportunistic construction process designed to overcome challenges with strictly top-down or bottom-up strategies. To address the three bridging challenges that arise during the construction process, Affinder also provides (2) an *unlimited vocabulary search* for discovering context features one may have forgotten; (3) *reflect and expand prompts* that help generalize one's concepts of the situation and expand one's efforts to forage for context-features; and (4) *simulation and repair tools* for identifying and resolving issues with how context features may operate on real use-cases differently than an author intends. We describe each of these functions below.

4.1 Block-Based Construction Environment

To overcome the shortcomings of top-down and bottom-up approaches, we draw on theories from opportunistic planning [20] to support an *opportunistic* construction process, in which a creator can follow both top-down or bottom-up processes at any time. Decisions and observations during construction may suggest new ideas or illuminate problems that cause the creator to shift their strategy. To support this construction process, Affinder employs a block-based programming environment that provides a single, visual workspace in which designers can (a) declare *concept variables* to represent intermediate concepts that serve as links between an abstract concept and available context features; (b) forage for context features by browsing and searching through categories of features; and (c) compose representations using logical operators; see Figure 3. We argue that this block-based approach can effectively support an opportunistic construction process by visually linking concept variables to context features; supporting recognition over recall; and reducing cognitive load by helping developers focus on concepts and how they are connected instead of on syntax and code.

Declaring concepts entails breaking down an idea about a situation into smaller concepts. These smaller concepts serve as a link between a conceptually-abstract situation and the available context features, and can be declared explicitly with *concept variables*. A creator can declare concept variables before defining their contents; this provides a visual reminder to compose building blocks later for the concept. For example, for a construction expressing 'situations to share a cheers', a user can represent a smaller concept 'drinking an evening beer' by declaring a concept variable, defining its contents using context-features and logical operators, and using the defined concept variable in a top-level concept variable named 'share a cheers'.

Foraging for features involves both searching and browsing for building blocks. Creators can either go in a top-down fashion by using their declared concepts to guide the types of building blocks they might look for, or a bottom-up fashion where they browse through the available features to see which ones might be relevant. A creator can use the *Toolbar* to navigate to building blocks based on categories such as weather, time of day, time of week, and time zone (Figure 3, Left of Middle). Creators can also use the *Search Interface* (Figure 3, Far Left) to find context-features based on a thousand place categories on Yelp. For example, a user expressing 'having an evening beer' could browse for features based on time of day and find a 'nighttime' block to drag into the work area; they might query the search interface for 'beer' to find relevant place context-features to add to the construction.

Composing Representations starts after creators find several context building-blocks which they combine together with logical operator blocks like *and, or, not.* Figure 3 shows an example of a logical composition used in the definition of 'drinking an evening beer' as various place contexts for having a beer ('barcrawl', 'tikibars', 'beer and wine').

4.2 Unlimited Vocabulary Search

Concept expressions can be underscoped in regards to the context-features a creator decides to include. This can occur when working with lengthy hierarchies of context-features (e.g., Yelp place categories), since designers can struggle to forage for features that can represent their intermediate concepts. To mitigate underscoping on detectable context-features, Affinder uses textual metadata from available APIs (e.g., reviews from Yelp) to create an unlimited vocabulary [17] of terms associated with the context features that users can query for based on their conception. This allows a creator to directly query for context features using aspects of a situation of interest, e.g., based on objects that afford actions, actions that can

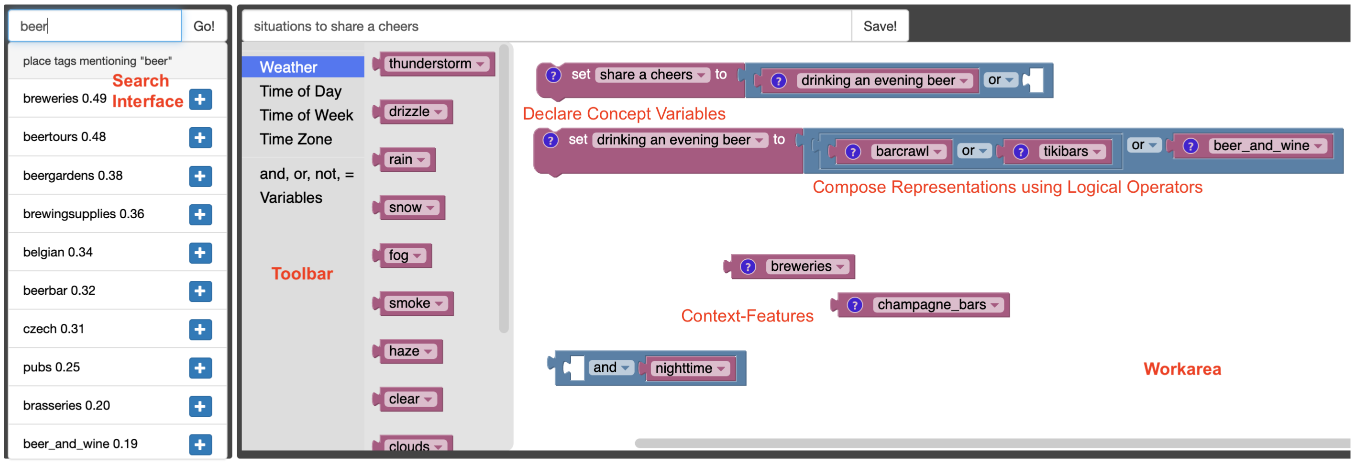


Figure 3: Affinder's *Toolbar* provides access to building blocks such as context features derived from weather, time, place in addition to logical operators like and, or, not, =. From the toolbar, users can drag and drop building blocks into the *Work Area* which is used to store relevant features and compose representations from building blocks. Affinder's *Search Interface* returns relevant place context-features. A button next to each 'adds' the feature to the work area.

be taken, activities that people are engaged with, etc. This helps developers discover context features they may have forgotten, and to broadly shift their notions of the concept they are trying to express and how to express it throughout the process of construction. For example, using Affinder's unlimited vocabulary search, the query 'field' could match the feature 'parks' through a review that says "Other aspects of the park include a people park with slides and swings, soccer *fields*, baseball *fields*, and plenty of open space;" or 'fields' could match the feature 'discgolf' through a review that says "The majority of the course is quite open and flat, playing around some decent sized trees, grassy *fields* and pedestrian pathways that are considered out of bounds."

4.3 Reflect and Expand Prompts

To mitigate underscoped concept expressions caused by design fixation on intermediate experience concepts, Affinder provides prompts that encourage users to *reflect* on generalizable concepts about why a context-feature is appropriate for the situated experience they are designing, and to *expand* their concept expression by foraging for context-features using this generalizable concept.

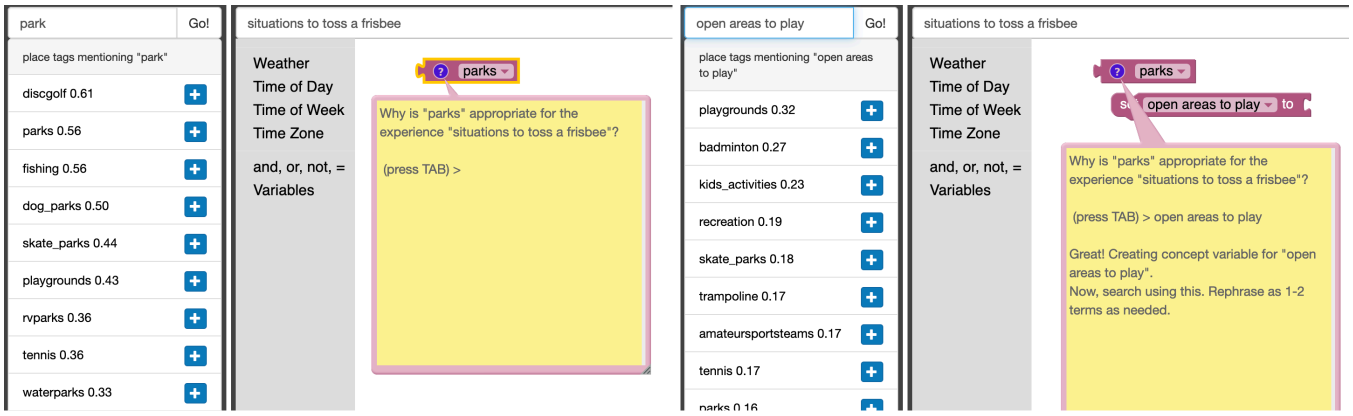
Each context-feature or concept variable can be used as the source of the reflection; reflection prompts can be activated for any of these by clicking on the corresponding blue question mark; see Figure 4. For example, on the left-side of Figure 4, a user has added the context-feature 'parks' in their work area, and chooses to reflect by pressing the '?' button associated with the 'parks' context-feature. The reflection prompt asks them the question *"Why is 'parks' appropriate for the experience 'situations to toss a frisbee'?"*. On the right-side of Figure 4, the user has proceeded to answer by typing 'open areas to play'. Upon pressing the tab key, a new concept variable is created that represents this generalized notion. They can subsequently use the search interface to find context-features matching the concept 'open areas to play'. We designed the '?' reflect button to be always visible and attached to the context-features in the workspace after observing during early testing that authors would sometimes forget that the reflect button was a feature they could use.

4.4 Simulation and Repair Tools

Given the challenge of knowing how a place context-feature applies to real-world cases based only its name, Affinder implements a feature for viewing example locations for a context-feature of interest (e.g., a list of the top 20 example locations tagged with the context-feature 'active' in Chicago), so designers can form a more accurate mental model of how a place category is used by the Yelp context-provider. This can help answer questions about context-features which are named in unexpected ways, such as what types of places the context-feature 'active' refers to.

Beyond viewing example locations for *individual* context-features, the full version of Affinder provides features for (A) simulating the execution of concept expressions composed of *multiple* context-features to help designers analyze real-world cases and label any that inaccurately represents the concept variable (as shown in Figure 5, Left); and (B) supporting designers to repair their concept expressions, so as to resolve outstanding issues in the execution of the concept variable (as shown in Figure 5, Right).

Affinder's Repair Tools supports two methods for repairing a concept expression so they operate more accurately: (1) using logical operators to exclude specific context-features; and (2) discarding features that are too inaccurate to be useful. First, context-features may not precisely match a concept (e.g., public gardens are parks but are not good for frisbee tossing). To resolve this problem, the first method of repair supports designers in excluding specific context-features (e.g., define 'open spaces to play' as 'parks' that are also not 'gardens'). By doing this repair, a designer can still effectively add a context-feature to their construction to increase coverage (e.g., most parks are good for frisbee tossing), while ensuring that this addition



place tags mentioning "park"

discgolf 0.61
parks 0.56
fishing 0.56
dog_parks 0.50
skate_parks 0.44
playgrounds 0.43
rvparks 0.36
tennis 0.36
waterparks 0.33

Why is "parks" appropriate for the experience "situations to toss a frisbee"?

(press TAB) >

place tags mentioning "open areas to play"

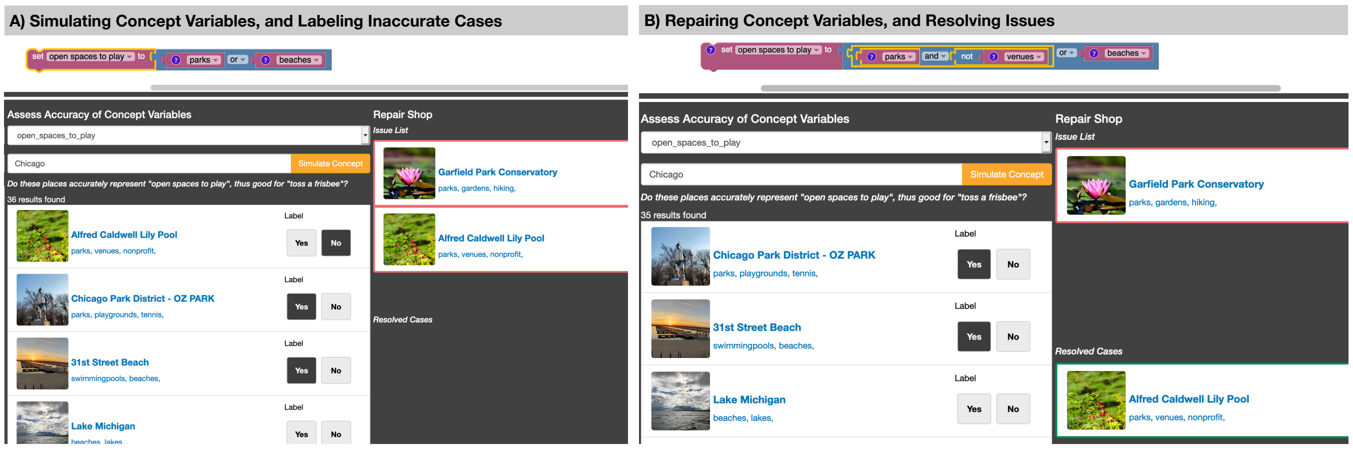
playgrounds 0.32
badminton 0.27
kids_activities 0.23
recreation 0.19
skate_parks 0.18
trampoline 0.17
amateursportsteams 0.17
tennis 0.17
parks 0.16

Why is "parks" appropriate for the experience "situations to toss a frisbee"?

(press TAB) > open areas to play

Great! Creating concept variable for "open areas to play".
Now, search using this. Rephrase as 1-2 terms as needed.

Figure 4: A series of two screenshots illustrates how a user might use the reflect and expand prompts to create new generalizable concept variables, and expand their efforts to forage for features.


A) Simulating Concept Variables, and Labeling Inaccurate Cases
set open spaces to play to
parks
or
beaches
Assess Accuracy of Concept Variables
open_spaces_to_play
Chicago
Simulate Concept
Repair Shop
Issue List
Resolved Cases
Label
Yes
No
B) Repairing Concept Variables, and Resolving Issues
set open spaces to play to
parks
and
not
venues
or
beaches

Do these places accurately represent "open spaces to play", thus good for "toss a frisbee"?

36 results found

Alfred Caldwell Lily Pool — parks, venues, nonprofit, — Label: Yes [✓ No]

Chicago Park District - OZ PARK — parks, playgrounds, tennis, — Label: [✓ Yes] No

31st Street Beach — swimmingpools, beaches, — Label: [✓ Yes] No

Lake Michigan — beaches, lakes, — Label: Yes No

Issue List: Garfield Park Conservatory — parks, gardens, hiking,; Alfred Caldwell Lily Pool — parks, venues, nonprofit,

Do these places accurately represent "open spaces to play", thus good for "toss a frisbee"?

35 results found

Chicago Park District - OZ PARK — parks, playgrounds, tennis, — Label: [✓ Yes] No

31st Street Beach — swimmingpools, beaches, — Label: [✓ Yes] No

Lake Michigan — beaches, lakes, — Label: Yes No

Issue List: Garfield Park Conservatory — parks, gardens, hiking,

Resolved Cases: Alfred Caldwell Lily Pool — parks, venues, nonprofit,

Figure 5: Designers can simulate concept expressions composed of multiple context-features (e.g., 'parks' or 'beaches' are 'open spaces to play'). Simulation can reveal cases in which context-features may fail to accurately represent specific concept variables (e.g., some 'parks' include locations like a conservatory and a lily pool which are not 'open spaces to play'). After labeling these inaccurate cases, designers can repair concept expressions to make better use of existing context features (e.g., parks only if they are also not 'gardens' or 'venues'). Simulating the repaired concept moves the offending case to a list of resolved cases.

does not sacrifice precision (e.g., so that special parks that don't fit 'open grassy area for play' concept would not be identified for the situation or activity). To help designers quickly identify cases that would require this type of repair, the list of detected cases is ordered by cases that are tagged with multiple context-features (a conservatory tagged 'parks', 'gardens', and 'hiking') appear before cases tagged with a single context-feature (a beach tagged 'beaches').

Second, a context-feature may be too inaccurate to be useful (e.g., 'recreation' mostly refers to indoor recreation centers and gyms which are not appropriate for frisbee tossing). Thus, the second method of repair allows a designer to discard this context-feature from the construction. By recognizing when a context-feature is too inaccurate and making the choice to discard it entirely, designers are able to assess when a context-feature they've added to their construction would inaccurately represent the high-level concept they were trying to express, and ultimately degrade the precision of their expression.

5 IMPLEMENTATION

Affinder is a Meteor.js web application[1] that uses Google's Blockly library [16] for its block-based construction interface. Affinder uses Blockly to generate a concept expression's corresponding Javascript code (e.g., (parks || beaches) && !windy;) which is a logical predicate that can be used by context-aware services to check whether the user is in a situation that matches the concept expression, or to produce a list of situations and locations that would. Applications can integrate this generated code by requesting

[1]https://github.com/NUDelta/affinder

a user's current context-features (e.g., Is the Yelp place category 'park' currently detected for a user? Is the user's current detected weather 'windy'?) from a context-provider API and evaluating the predicate. Affinder's simulate and repair tool uses the generated code to simulate the concept expression on real-world place venues.

Affinder's unlimited vocabulary search engine was built by applying the term-frequency inverse-document frequency (TF-IDF) statistic to a corpus of 1241 documents corresponding to Yelp place category context-features. Each document comprised community-authored reviews for a single place category (e.g., 'parks') across all listed places in 8 major metropolitan areas [23]. The reviews associated with a place venue which is tagged with multiple place categories (e.g., a public park tagged as 'park', and 'playground') will be included in the text documents associated with all place categories. During iterative prototyping and testing, we observed that this also had the desirable side-effect of broadening the set of returned place category context-features. Document relevance for a query was based on the sum of TF-IDF values for all terms in a query [35], and the 25 top place categories are returned.

Affinder's feature for simulating and repairing concept expressions uses the Yelp Fusion Business API [24] to return a list of real-world places for a given city. Our implementation of simulating a concept expression creates a list of locations by taking the set union of the top 20 locations for each of the context-features in an expression (e.g., a concept expression open spaces = parks || beaches has two context-features and will return 40 real-world locations for a target city). Then, the set of positive predictions is obtained by applying the concept expression's corresponding Javascript code to the list of location venues.

6 USER STUDY

We performed a comparison study to evaluate the extent to which Affinder supports creators in effectively encoding their human concepts of a situation into a machine representation using available context-features. Specifically, we conducted a between-subjects study that compared authors' use of the full version of Affinder with all the features for overcoming the bridging challenges vs. a baseline version of Affinder with a reduced set of features.

In this study we ask: **RQ1**: Does unlimited vocabulary search help designers find relevant context-features, to overcome the challenge of a concept expression being underscoped? **RQ2**: Do reflect and expand prompts help designers stretch their concepts and efforts to forage for context features? **RQ3**: Do simulation and repair tools help designer's recognize cases when a concept expression does not operate as intended on real-world cases, to overcome concept expression inaccuracies?

6.1 Method and Analysis

6.1.1 Experimental vs. Baseline Versions. For this between-subjects study, we provided participants two versions of Affinder: an experimental version with all the core features for overcoming the specific bridging challenges (unlimited vocabulary, reflect and expand prompts, simulation and repair tools), and a baseline version without these features. Both versions support an opportunistic construction process through its block-based environment where an author can follow a top-down or bottom-up process at any time; and both versions use the same set of base context-features including Yelp place categories, time, and weather features.

However, the baseline does not include the core features that address specific challenges arising within the bridging problem.

(1) Instead of the unlimited vocabulary search, authors forage for place category context-features using a simple text search (e.g., searching 'park' will match context-features 'parks' and 'parking lots', whereas searching 'frisbee' returns no place categories). We hypothesize that without unlimited vocabulary, authors using the baseline will struggle to access relevant features based on their conceptions because they will not know the exact names of the place context-features.
(2) Authors using the baseline do not have explicit prompts for reflecting and expanding concept expressions. Without these prompts, we hypothesize that users may fixate on their early concepts of a situation, and thus have conceptually-narrower searches for context-features.
(3) Authors using the baseline cannot simulate how their entire concept expression operates on real-world cases, nor keep an issue list to guide the repair of expressions. Instead, authors are only provided the tool that allows them to view examples of place venues for a single place category. This allows users to clarify the usage of any context-features they are uncertain about, such as ones that are named in unexpected ways (e.g., what kinds of locations would be tagged on Yelp as the place category "active"?). We hypothesize that without the full set of simulate and repair tools, authors will miss cases where their concept expression operates differently than they intended, which will result in concept expression inaccuracies on real-world cases.

6.1.2 Participants. We recruited 14 participants from several undergraduate HCI classes from a mid-sized university in the Midwestern US. Everyone had some prior background in designing computing technologies. While participants were not specifically selected to have a background in developing location-based or context-aware computing applications, we provided them with sufficient background material on the potential uses of our application and how our application would detect end-user contexts; we describe the details of our background and tutorial procedure in the next section. Each participant was compensated with a $25 gift card for their participation in the 1 hour long study. For this between-subjects study, 8 of the participants were assigned to use the version of Affinder with all the technical features (unlimited vocabulary search, reflect and expand prompts, simulation and repair tools), and the other 6 were assigned to use the baseline version without all the technical features.

6.1.3 Study Procedure. Participants were asked to watch a 5 minute background video prior to joining a video conference study session. In this video, they learned about our vision for a context-aware, social application use-case in which the app helps friends engage in a digitally-mediated shared experience when they are in similar situations at distance [32]. We described an example scenario of how a friend in Chicago having an evening beer and another friend in Taiwan drinking a morning tea could use the application to share digitally-mediated cheers with their beverages. Participants were told they would act as a designer of this context-aware application that uses mobile context-features (e.g., place, weather, and time) to detect aspects of a situation that would support engaging in these shared experiences. Specifically, they were told that using Affinder, they would figure out (1) what the possible situations are where people can engage in the shared experience (e.g., raise their beverages for a digitally-mediated cheers) and (2) how to express these situations using context-features like Yelp place categories.

They were then given a guided, hands-on tutorial creating a concept expression for "situations to share a cheers" using the version of Affinder that they were assigned to (15 min); as the experimental version had more features to teach, their tutorial usually took longer. After the tutorial, users constructed up to two concept expressions for situations to engage in shared experiences, as time allowed (30-35 min). These situations included 'awesome situations to toss a frisbee' and 'situations for grabbing food that is good for a cold day'.

With the full feature version, we expected users to expand their notion of the concept they were expressing; therefore, these participants naturally spent more time on the task, and often had time to complete only one concept expression. With the baseline, participants tended to naturally run out of ideas and complete their concept expression early; thus, participants often had time to complete two concept expressions in the time available. We observed and recorded the participant's construction process, and asked them to talk-aloud to explain their decisions while creating the situation detectors. Finally, they completed a post-study questionnaire and a semi-structured interview (15 min).

6.1.4 Measures and Analysis. Our answers to the research questions are triangulated amongst 3 sources of data: (1) qualitative descriptions and summary statics about the concept expression, captured as it evolved during the construction process and once an author has completed it; (2) qualitative observations of authors' behaviors and usage of Affinder's features; and (3) qualitative insights about participants thoughts and strategies during their construction process. We opted to use this qualitative approach because we can evaluate the core features of Affinder together in one interface while still gaining insight into how using each of the core features make a difference in how the concept expression evolves and how an author's thoughts and strategies change.

To summarize the created concept expression, we measured the breadth of context-features included in the concept expressions by counting the number of relevant place context-features included.

To answer RQ1, we noted search queries that were made and which relevant context-features were added to the concept expression while using the unlimited vocabulary vs. the simple text search. Additionally, we used talk-alouds and retrospective interviews to understand (1) how the context-features a participant saw in the search results influenced their thoughts, and (2) how these updated thoughts led to foraging and adding additional context-features.

To answer RQ2, we asked about moments when users activated the reflect and expand prompts. This included details about (1) the initial idea they chose to reflect on; (2) the general concept they articulated that made their initial idea appropriate for the situation; (3) how the general concept later influenced their thoughts as they talked-aloud; and (4) what actions they did shortly afterward, such as unlimited vocabulary searches that followed or how their concept expression evolved.

To answer RQ3, we looked for cases when users, after simulating their concept expressions, updated their concept expressions. Through talk-alouds and revisiting these cases in the interview, we gained a better understanding of (1) a user's prior notions for how the intended a concept expression to operate; (2) what real-world case they found that posed an issue or surfaced a misconception in how the concept expression operated; and (3) how they decided to repair the concept expression through removing or explicitly negating specific context-features.

7 RESULTS

Across both versions of Affinder, our 14 participants (8 for experimental, 6 for baseline) created a total of 20 concept expressions (10 for experimental, 10 for baseline). Figure 6 shows two constructions that were made using the version of Affinder with all of its core features. An example construction expresses 'grabbing food for a cold day' as any places serving hot beverages (tea, coffeeshops, coffee), food with soup (hotpot, ramen, soup), or spicy food (thai, japanese curry, bbq, szechuan) and where the weather is cold. Another example construction expresses 'places to toss a frisbee' as either open outdoor public spaces ('parks', 'beaches', or 'playgrounds') where the weather is clear and it is daytime, or open indoor public spaces (including most 'gyms' or 'recreation' but excluding cases that support outdoor activities, e.g., 'hiking'; or activities associated with small spaces e.g., 'boxing'). In both of these example cases, participants used Affinder to flesh out their concepts of the situation and bridge to a wide range of detectable context-features. Participants switched between top-down and bottom-up processes, allowing their concepts to evolve while foraging for the lower-level context-features. For example, a participant expressing "situations to throw a frisbee" started by foraging for the first places that came to mind such as parks and beaches; after declaring a concept variable for "outdoor public spaces" to unify these place context features, they realized that several outdoor and indoor places supporting athletic activities could also work, thereby expanding their efforts to forage for a wider range of place contexts.

In contrast, Figure 7 highlights two example constructions that were made using the baseline version of Affinder. One example construction expresses 'grabbing food for a cold day' as when the weather is cold and user is either eating soup (restaurants with 'soup'), drinking hot chocolate or coffee ('cafes', 'hong kong cafes', 'themed cafes'), drinking tea (places serving 'tea'), but not eating ice cream (places serving 'ice cream'). Another example construction expresses 'situations to toss a frisbee' as places with open fields (parks or beaches), where there is warm weather (clear or hot), and it is daytime. Similar to participants using the experimental version, participants using the baseline were able to create concept variables that linked their concepts of the situation to detectable context-features. In addition, participants were also able to move between top-down and bottom-up processes. For example, a participant started by foraging for place context-features that matched their initial concept of "drinking hot chocolate" and found many context-features related to cafes; then, they created more concept variables

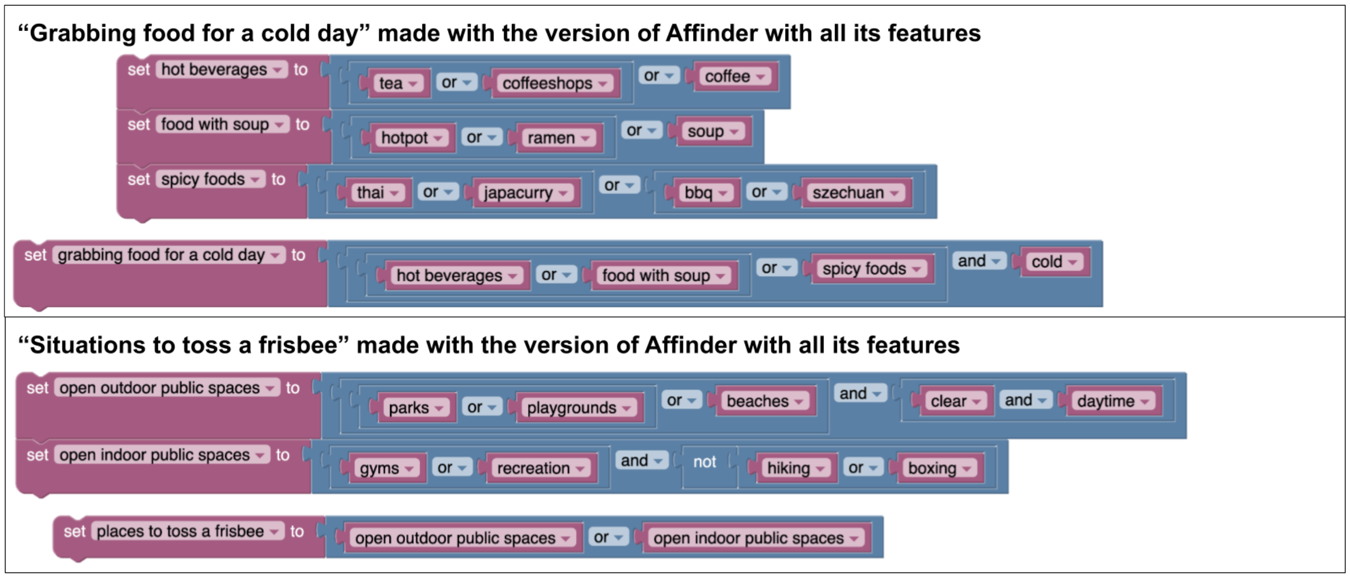

"Grabbing food for a cold day" made with the version of Affinder with all its features
set hot beverages to tea or coffeeshops or coffee
set food with soup to hotpot or ramen or soup
set spicy foods to thai or japacurry or bbq or szechuan
set grabbing food for a cold day to hot beverages or food with soup or spicy foods and cold
"Situations to toss a frisbee" made with the version of Affinder with all its features
set open outdoor public spaces to parks or playgrounds or beaches and clear and daytime
set open indoor public spaces to gyms or recreation and not hiking or boxing
set places to toss a frisbee to open outdoor public spaces or open indoor public spaces

Figure 6: Two concept expressions made with the full version of Affinder with all its features. The first construction expresses 'grabbing food for a cold day' as having hot beverages (tea, coffeeshops, or coffee), spicy foods (thai, japacurry, bbq, or szechuan), or food with soup (hotpot, ramen, or soup) and while the weather is cold. The second construction expresses 'situations to toss a frisbee' as open outdoor public spaces (parks, beaches, or playgrounds, and when it is clear and daytime) and open indoor public spaces (most gyms or recreation, but excluding cases with hiking or boxing.)

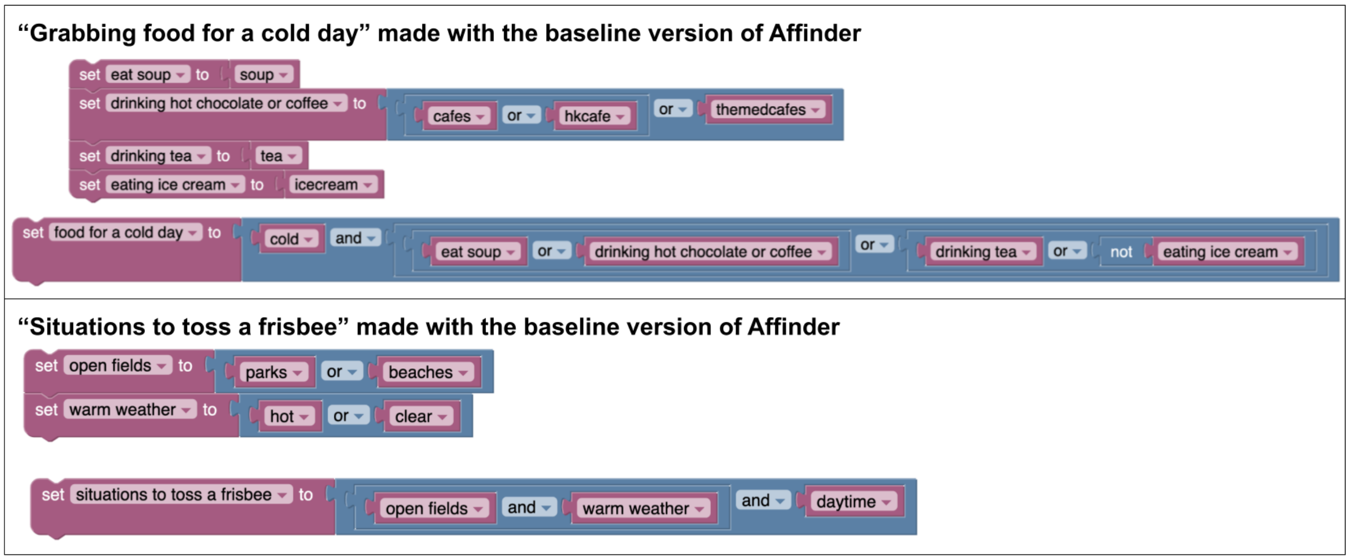

"Grabbing food for a cold day" made with the baseline version of Affinder
set eat soup to soup
set drinking hot chocolate or coffee to cafes or hkcafe or themedcafes
set drinking tea to tea
set eating ice cream to icecream
set food for a cold day to cold and eat soup or drinking hot chocolate or coffee or drinking tea or not eating ice cream
"Situations to toss a frisbee" made with the baseline version of Affinder
set open fields to parks or beaches
set warm weather to hot or clear
set situations to toss a frisbee to open fields and warm weather and daytime

Figure 7: Two concept expressions made with the baseline version of Affinder without all of its core features. The first construction expresses "grabbing food for a cold day" as when the weather is cold and a user is eating soup, drinking hot chocolate or coffee (cafes, hkcafe, themedcafe), drinking tea, or not having icecream. The second construction expresses "situations to toss a frisbee" as open fields (parks,beaches) and when it is warm weather (clear and hot) and daytime.

for other hot foods one might find at cafes like "tea" and "soup" and foraged for context-features matching these.

However, authors using the baseline declared concepts which were closer to the detectable context features (e.g., drinking tea) as opposed to more general concepts that would unify across context-features (e.g., hot beverages). As such, these concepts variables were often linked to one or two place context-features (e.g., eating soup as 'soup'). For participants who did declare more general concept variables (e.g., open fields), they often struggled using the baseline to forage for a wider range of context-features matching these concepts.

We measured the concept expression for the breadth of context-features included in their constructions. For the concept expression ‘grabbing food for a cold day’, the median construction had 11 place context-features made with the full version of Affinder, while the median construction had 8 context-features made with the baseline version. For the concept expression ‘awesome situations to toss a frisbee’, the median construction had 4.5 place context-features when made with the full version of Affinder, while the median construction made with the baseline had 2 place context-features.

Having described an overview of the concept expressions created using both versions of Affinder, we now turn our attention to describing how each of the core technical features (unlimited vocabulary, reflect and expand prompts, and simulation and repair tools) supported designers' construction processes.

7.1 Results for Unlimited Vocabulary Search

All 8 participants in the experimental condition used Unlimited Vocabulary Search to forage for context-features from the Yelp Places API that matched their initial concepts. For example, P7 tried to represent the concept of ‘snowy environments’ as a type of situation for ‘grabbing food on a cold day’. By searching for terms such as ‘ice rinks’ and ‘snow day’, they were able to find several place context-features matching their concept (skatingrinks, skiresorts, skischools). P1 said: *“I liked to see other examples of other [place contexts], I didn't have to think about all these other [place contexts] on my own.”*

In contrast, participants using simple text search struggled to forage for context-features matching their conceptions. This happened because the concept vocabulary users formed as search queries (e.g., ‘spicy’, ‘grass’, ‘meadows’, ‘frisbee’) returned few or no matching context-features. For instance, P7 expected various sport fields to show up after searching ‘fields’, but ‘baseballfields’ was the only result. In contrast with the experimental condition, P9's search for a similar concept, ‘soccer field’, using the Unlimited Vocabulary Search helped them find ‘playgrounds’ and ‘college universities’ as relevant context-features.

Beyond finding detectable context-features matching their conceptions, participants using Unlimited Vocabulary Search were able to stretch and update their own concepts of the situation they were trying to express. This happened while browsing through the context-features in the search results that were associated but not what they were expecting to find. We present some examples from our user study where the search engine supported stretching concepts (see Figure 8). For example, when asked how the unlimited vocabulary search was helpful, P9 explained: *“What comes up in the [unlimited vocabulary search] suggestions already gives you ideas. For example, after seeing soup, I shifted my thinking to soup based queries, like pho and other soup noodle stuff... If I see something related, but in a different way, it helps me generate more [concepts] that I want to search.”*

In comparison, for participants using simple text search, the absence of related context-features led to stopping in their search for place context-features and expressing their concepts. For instance, when P6, a participant in the baseline condition, was asked how they decide when they have completed constructing, they said, *“When I run out of tags to search for... after finding ‘parks’ and ‘playgrounds’, grass and lawns wasn't showing me anything. Then I run out of ideas.”*

7.2 Results for Reflect and Expand Prompts

Reflect and Expand prompts were used by 7 of the 8 experimental participants across 12 moments to articulate more generalized concepts and create new concept variables. 5 of the 7 participants used the articulated concepts to expand their efforts to forage for new context-features. We show in Figure 9 several examples of how reflection prompts led to expanded foraging efforts. 4 of these moments lead to successfully discovering new context-features that were added to their concept expression; while in 2 other moments, designers attempted to use the generalized concepts to search, but were ultimately unsuccessful at discovering new features.

As one example of how reflection prompts led to adding new context-features to a construction, P5 reflected on why their idea of hot chocolate was appropriate for ‘grabbing food on a cold day’. Upon reflection, they realized that during winter holidays, “hot chocolate” is good with other sweet, hearty baked goods – implying that “hot” food isn't the only attribute, but any sweet foods that remind of winter would also be good to eat on a cold day. After P5 answered the reflection prompt, thereby creating a generalized concept variable named ‘sweets to eat’, they began to forage for place context-features matching this concept. By using searches such as ‘desserts’ and ‘bakery’, they were able to add ‘desserts’ and ‘bakeries’ as additional place context-features to the construction.

However, for 2 participants, foraging for context-features using the generalized concept variable did not find relevant context-features. For example, after P12 articulated how ‘parks’ are appropriate ‘situations awesome to toss a frisbee’ because they are ‘outdoor public spaces‘, they could not find relevant features when searching for ‘open public spaces‘ (which returned contexts like ‘sharedofficespaces‘, ‘libraries‘, ‘publicservicesgovt’, ‘galleries’, ‘communitycenters’). They tried to revise their query by searching ‘outdoors’, but did not find any other context-features they did not already have in their construction. This suggests that the generalized concept variables created through reflection may not always be useful when used verbatim as search queries, which may require redesigning the prompts to guide users to not only articulate the general reasons for appropriateness, but also to form useful queries from these concepts that are aligned with the underlying textual metadata (e.g., how others describe Yelp reviews). Additionally, users of the unlimited vocabulary may need support in understanding why their query is not returning results they are expecting, and how they might revise their query to more effectively bridge to the relevant context-features.

Some users used the prompts to reflect and create generalized concept variables, but did not aim to expand their foraging efforts using these generalized concepts. Rather, 2 of the 7 participants who activated the reflect and expand prompts created a generalized concept variable that unified context-features that they had already added to construction workspace. For example, P9 reflected on why ‘szechuan’ was appropriate for ‘situations to grab food on a cold day’, then created the concept variable ‘spicy foods’, and finally

search term	previous notion	feature that inspired updates	updated notion	behavior that followed
spicy	general notion of spicy food to warm the body	thai	curries in thai is a great example of spicy warm food for cold day	searched `curry' and added the feature `japacurry'
open space outdoors	physical outdoor locations with open space	active	places that support being physically active might be a good space for tossing a frisbee	searched `exercise' and adds the feature `bubblesoccer'
hot	hot chocolate and other hot food is good for a cold day	hotpot	asian soups are particularly good for cold days	searched `ramen' and added the feature `ramen'

Figure 8: Examples of queries made with the unlimited vocabulary search that helped designers discover context-features that inspired updates to their notions of their concepts. After a designer had shifted their notions, they often continued searching based on their updated notions and added new features.

Idea to Reflect On	Articulated Concept of Appropriateness	Searches that Followed	Relevant Concepts Added to Construction
hot chocolate	hot chocolate is a type of 'sweet' which is consumed during the cold, winter holidays	desserts, bakery	sweets to eat (desserts, bakeries)
coffee	coffee is an example of a 'hot beverage'	latte, mocha	hot beverages (tea, coffee, coffeeshops)
parks	some parks have vendors where you can buy and eat food outdoors; similarly, an 'outdoor festival' or farmers market with vendors selling food and drinks is perfect for eating outdoors	outdoor festivals, mead	outdoor festivals (cideries, wineries, fleamarkets)

Figure 9: Examples of how Reflect and Expand prompts helped users reflect on an idea part of their construction, articulate concepts about what makes that idea appropriate, and expand their foraging efforts by subsequently searching and adding new concepts and features to their construction.

used a logical ‘or’ operator to unify existing features like ‘szechuan’, ‘japacurry’, and ‘thai’.

We sought to understand when and why users activate the reflection prompt during their construction process. Some participants used the reflection prompts during moments when they themselves felt stuck. Participants often felt stuck when they could not find additional place context features after trying several search queries. For example, as P7 was trying to express the situation “food for a cold day”, they felt stuck and fixated: “*my mind keeps going to snow days, and from the standpoint of that situation, I think I got [all the context-features] out of that. So I think I should use a reflection prompt.*” Others participants understood the reflection prompts as a primary way to create concept variables, and used the reflect button to create concepts that unified existing context-features: “Once I equated the blue question mark as a way to create [a concept] that linked [context-features] together with a logical expression, I developed that association... and used it when I wanted to create a general category for the features I had added” (P13). Overall, authors initiated the prompts for reflecting and expanding their concepts during transition points in their process when they felt they had stopped on their current task to forage for context-features. In this way, its usage aligns with how Affinders was designed to allow authors to move flexibly between foraging for context-feature matching concepts (top-down) and fleshing out concepts that could link features (bottom-up) when issues or decisions arise during the construction process.

7.3 Results for Simulate and Repair Tools

Affinder’s simulate and repair capabilities helped participants identify issues with how their concept expressions operated in real-world locations, and refine concept expressions to be more accurate. We present some examples from our user study where participants used simulation to find issues in real-world cases and refine concept expression to resolve issues; see Figure 10. Participants updated their concepts when they saw items from simulation different from their original mental image, helping enrich their understanding of the nuances of how context-features actually apply to real-world cases. As an example of this process (see Row 1 in Figure 10), P14 was originally thinking of ‘baseball fields’ as a ‘place for throwing a frisbee’, but recognized through simulation that some baseball fields are categorized as ‘stadiums and arenas’, which would make it harder to access the grassy outfield. Next, P14 used the repair shop to update their concept expression for ‘open space with grass’ to include baseball fields that are not also stadiums and arenas.

In another example (see Row 2 in Figure 10), P3’s original construction expressed ‘big open space’ as parks or beaches. Through simulation, they realized that some places, like a Conservatory or Lily Pool, are meant for formal events and activities and thus would make for an inappropriate location to throw a frisbee. Next, P3 used

Construction Before	Prior Notions	Misconceptions or Issues found in Real-World Cases	Construction Afterwards
open_space_with_grass = (campgrounds or baseballfields)	baseballfields in a park where the grassy outfield is open to use campgrounds will have open grassy spaces to throw a frisbee	a place tagged with baseballfields and stadiumsarenas will be more restrictive and less appropriate to use the outfield if a campgrounds support activities like rafting, the physical environment will likely not have open grassy areas	open_space_with_grass = (campgrounds **and (not rafting)**) or (baseballfields **and (not stadiumsarenas)**)
big_open_space = (parks or beaches)	open space at a public park	there are some parks, like conservatories or lily pools that are tagged as gardens or venues, that are too fancy and inappropriate as places to toss a frisbee	big_open_space = (parks or beaches) **and (not (gardens or venues))**
hot_drinks_and_pastries = (bakeries or coffee or coffeeshops or coffeeroasteries or cafes or tea)	bakery cafes are where you can typically sit to have a coffee/hot chocolate with the pastry	a bakery in a grocery store does not support the same leisurely eat and drink experience	hot_drinks_and_pastries = (bakeries or coffee or coffeeshops or coffeeroasteries or cafes or tea) **and (not intlgrocery)**

Figure 10: Examples of Affinder's Simulate and Repair Tools helped users simulate their concept expression, identify real-world cases that highlight misconceptions and raise issues in precision, and repair their concept expressions.

the Repair Shop's Issue List to notice that the Conservatory was tagged as a 'garden' and 'park', while the Lily Pool was tagged as a 'venue' and 'park'. To resolve these issues, the designer updated their construction to express 'big open space' as parks or beaches that are not also gardens or venues.

We were interested in how usage differed between simulate and repair capabilities in the full version of Affinder versus the feature for viewing example locations for a single place context-feature. Many participants simulated concept expressions once they had completed their representation for a concept variable. For example, P12 used the simulate feature composing together context-features for a 'warm foods enjoyed indoors' concept; they said: *"when I simulate my concept I am trying to find [location] options that don't fit. But I'm looking and there's really nothing that stands out."* When designers did find location options that did not match their concept, they would label these issue locations and proceed to the repair shop to refine the concept expressions to resolve the issues. In this way, the tools for simulating an entire representation helped users to complete a representation of a concept variable, simulate to see if any issues arose for any of the multiple context-features, and resolve any of these issues as they arose.

In contrast, a majority of participants used view example locations to understand a specific context-feature they were uncertain about. For instance, P11 thought the context-feature 'active' might be relevant for tossing a frisbee, but did not know what this context-feature meant based on its name; viewing example places helped them understand that 'active' contained many indoor gym and activity centers that were irrelevant for the 'spacious and nature' concept they were trying to express. In another case, P14 chose to view example places of 'bbq' because they thought that it would apply to locations where one can use a grill outside to barbecue, which is inappropriate for 'good situations to grab food on a cold day'. Instead, the examples were barbeque restaurants, which they felt matched the situation they were expressing. In general, viewing example locations supported a designer's while they foraged for features, helping them understand whether any individual feature they felt uncertain should be included in the expression.

However, view example locations can only identify misconceptions for context-features a participant chooses to inspect. For example, P11 said: *"I didn't need to view example places for 'campgrounds' and 'baseballfields' because I already know what those types of places are."* This suggests that if participants feel certain about the context-features, they may not choose to view example places for those features. However, this same participant later simulated the entire concept expression, and found nuances in how a context-feature belonging to the expression applies, such as how 'campgrounds' that support 'rafting' would not apply to their concept because such locations would have more rivers instead of open grassy areas required for tossing a frisbee. This suggests that simulating and refining a concept expression after they have completed it serves a role in surfacing inaccuracies in how a context-feature detects real-world places, without designers having to explicitly look for them.

8 DISCUSSION

Having demonstrated how Affinder can help authors express their concepts of a situation to machines, we revisit the core ideas behind Affinder, discuss how they may generally enable the expression layer between human concepts and machine representations, and envision the role of expression tools for developing intelligent applications that facilitate human activities and experiences.

8.1 Enable the Expression Layer between Human Concepts and Machine Representations

In this work, we uncovered a set of bridging challenges that arise when designers of context-aware applications express conceptually rich situations that are several layers of abstraction removed from the underlying machine features. Affinder's design principles are distinguished from most other context programming tools, as they explicitly support human cognition when working with context-features. To support the bridging between human concepts of a situation and machine representations, Affinder supports two interconnected cognitive processes. First, Affinder helps designers flesh out and expand their concepts for how a situation affords engaging in a human activity or experience. Second, Affinder helps designers link their concepts to machine-detectable context-features.

8.1.1 Fleshing out and Expanding Human Concepts. We found that expanding one's conceptual notions was important for helping designers explore the conceptual space and overcome challenges with underscoped concept expressions. Designers who used the reflect and expand prompts stretched their concepts of a situation by reflecting on why a context-feature they added was appropriate for the situation they were trying to express; this process of using the prompts inspired new searches, ultimately helping with overcoming underscoping of concepts. Additionally, designers who used the unlimited vocabulary search tool encountered context-features that shifted their notions of the situation they were trying to express; these updates led to expanding their search queries for additional context-features.

The core ideas behind these two techniques for conceptual stretch have parallels to existing research tools for conceptual ideation and overcoming design fixation. First, Affinder's reflect and expand tools support conceptual stretching by scaffolding authors to re-represent their initial ideas at a higher-level of abstraction, to help designers remember other contexts that might also apply. This core idea adapts methods for re-representing general linguistic terms to increase the chances that people recall additional ideas from other parts of the conceptual space [28]. This literature uses these methods to help product designers remember useful analogs, i.e., solutions that can be adapted from other domains to solve their current design problem. In contrast, we use it to support context-aware application designers to re-represent their human concept of a situation at different levels of abstraction, so that they can find multiple ways of expressing it using context-features.

Second, Affinder's search tools led to conceptual stretching in several cases by helping with the discovery of place contexts that were conceptually different from their initial concepts. Within HCI, several creative ideation tools have also supported the search and discovery of example ideas for the purposes of inspiring and influencing a designer's concepts [38, 39, 41]. Many of these tools for supporting conceptual stretch have been isolated to the concept-ideation phases of the design process, where designers are sketching a description or image of a product design idea. In contrast, with Affinder, searching for conceptually-inspirational example items is interconnected with other stages of the design process like finding implementable forms for concepts—as the unlimited vocabulary search tool can serve the dual-purposes of stretching a designer's concepts, and helping them find matching context-features for an existing concept.

Supporting cognitive processes like conceptual stretching will be important for other design or creative processes in which ideation and implementation are interconnected processes. One such creative process is in the field of human-AI co-creation with generative models, where users must express their creative concepts through a collaboration with an AI capable of generating content. Within the domain of music co-creation, composers must flesh out their concepts for how to achieve a creative goal (e.g,. expressing the emotions of sad and stuck in the music), while also strategizing on how to implement their concepts through using the available controls to steer the AI to generate music that expresses their intent [31]. Studies of human-AI music co-creation have observed cases where a human creator will change their own concepts of how to express a human emotion through music, based on interacting with music alternatives generated by the AI [30]. This finding represents a case where the exploration of implementable solutions can inspire refinement of human concepts. In addition to example-driven conceptual shifts, such tools can encourage reflection and expansion of concepts through re-representing concepts at different levels of abstraction.

8.1.2 Linking Human Concepts and Implementable Machine Representations. In addition to coming up with concepts for solutions, Affinder also supports finding an implementable form–in that it helps designers translate their concepts into machine representations using detectable context-features. The final output is a piece of code that uses detectable context-features as input, and can be used in an intelligent context. The problem Affinder solves makes it distinguished from conceptual ideation tools for this reason. Affinder effectively supports this translation process by providing (1) a rich vocabulary in which authors can query the available set of machine features, and (2) tools for checking how a machine's features may or may not accurately represent a designer's concepts on real-world cases.

Affinder's core ideas have parallels with interactive machine learning paradigms, such as active learning and machine teaching. In this setup, a human evaluates and refines a machine representation, such as machine classification model, by testing it on new and existing cases and changing the machine's representations—either indirectly via providing labels to a machine learner [2], or more directly by specifying or removing features in the representation [6]—to improve the machine's ability to accurately identify the desired concept. Most of this work typically relies on humans to provide labels of a concept so that a learning algorithm can infer which machine features might be relevant to include or to put greater weight on. Affinder takes a different approach, allowing a human's concepts, externalized through natural language, to be used as queries for discovering which machine features could be useful. We argue that this is an effective and complementary approach by letting humans find useful machine representations through using a human's richer notions of the concept. This approach should be useful when the machine features are too numerous to browse through, but do have semantically-meaningful metadata that can be used for querying.

Bridging problems can manifest in other domains where humans must express their rich concepts into implementable machine-representations. As we have discussed, bridging problems are a marriage of the problems of conceptual ideation and finding implementable machine representations for concepts. As such, tools like Affinder which overcome bridging challenges need to embrace ideas from both literatures on design concept ideation and interactive machine teaching. We argue that the next generation of expression tools will closely support the processes of fleshing out concepts and linking to machine representations. There is a double loop between humans refining their mental concepts, and refining the machine representations too. In this way, construction processes, like the one Affinder aims to support, help to create expressions that are more human, while also more detectable.

8.2 Imbue machines with an understanding of human situations and the experiences and activities they afford

Two decades ago, Abowd and Mynatt envisioned a near future in which computing technologies would augment and benefit our everyday lives. Everyday computing would support a mode of continuous interaction where computing technologies were no longer just a localized tool, but a constant companion that runs in the background, and could act opportunistically to promote the informal and unstructured activities of our everyday lives, from orchestrating tasks, to communicating with family and friends [1]. At the same time, researchers began prototyping the types of context-aware applications to step towards this vision, largely enabled by technical advances in sensors and algorithms for inferring aspects of human context, as well as frameworks and toolkits that made it easier to write applications using sensors and component detectors [10]. Yet amidst the technical opportunities afforded by such advances, it became apparent that the importance of context-aware computing extended beyond the contextual factors that machines can detect (e.g., spatial location, user identity, proximity of people and devices). Equally important was considering how these contextual factors contribute to the meaningfulness of humans acting and relating in situations [14].

In some ways, these challenges still exist within the current landscape of technologies. The technologies we have today—including the variety of commercially-available physical and virtual sensors (e.g., cameras, smart home devices, location-based metadata, social media and application usage) and the machine learning and algorithms for processing this sensor data—are starting to give applications a richer understanding of people's everyday worlds. Yet, to really leverage these component detectors to create applications that have an awareness of our human ways of acting and relating in situations, application designers ultimately need to bridge between their human concepts of a situation and the machine's available detectors. That's where tools like Affinder and support for bridging becomes ever more important.

In the near future, as emerging technologies like augmented reality (AR) and virtual reality (VR) will likely take an increasing role in mediating our social interactions and personal activities within everyday contexts [21], a richer understanding of our human situations in these environments will be important for ensuring these technologies can be aware of and facilitate the experiences users want to have. Devices for AR, such as AR glasses and mixed reality headsets, will have access to the familiar set of context-features based on location, activity, and time that current mobile context-aware platforms provide—as well as additional context-features such as object-classes recognized through computer vision [25]. While these future technologies will be able to detect aspects about a user's context, such as the place they are located and or the objects in front of them, tools like Affinder and support for bridging will be required to imbue these applications with a deeper understanding of these human situations and the experiences and activities afforded in them. We anticipate that tools for expressing the human ways of relating and acting in situations will be important for facilitating user social experiences and activities occurring in VR as well. Collaborative virtual environments should have a sense of place, where there are social norms and understandings of what experiences or activities are appropriate in these environments [15, 19]; thus, systems that facilitate interactions in these virtual environments should have an understanding of human experiences and activities that can take place in these situations too. While some virtual places may borrow the social norms from the real-world physical places they are modeled after (e.g., a VR bar or cafe), it will be important that application designers have the tools to express their concepts of how virtual situations may differ from their physical situation counterparts, and what experiences and activities in virtual reality are afforded and appropriate in them.

9 LIMITATIONS AND REMAINING OBSTACLES

In our final user study, we did not recruit authentic designers with prior experiences creating context-aware applications. Instead, participants were novices from an undergraduate design and HCI curriculum. Accounting for this, rather than focusing on how authentic context-aware designers perceive the effectiveness of Affinder, we sought to understand how Affinder's core features can help overcome specific bridging challenges–which can arise for any designer, no matter their specific background. In future studies, we could gain feedback from authentic, context-aware application designers and better understand how they imagine tools like Affinder being useful for expressing situations for their application use-cases, and adapted as part of their own development workflows.

Affinder was built and tested for location-based contexts, such as place venues that one might encounter while being mobile across one's day. By leveraging Yelp place categories, Affinder is more useful for describing situations that could occur at places in a town or city such as parks or types of restaurants but currently less useful for describing situational contexts that occur within the home or office as detected by indoor sensors. To expand the set of detectable contexts, future developments of Affinder could include additional base context-features beyond Yelp place categories. When extending this context-set, it would be important to consider whether it is sufficient to create another toolbar list that a user can browse through, or whether techniques like the unlimited vocabulary search would be needed to help users discover context-features. For example, for context-features describing different areas of a home (e.g., dining room, kitchen, bathroom, garage, backyard), it may be sufficient to list them in Affinder's toolbar for users to browse through.

One core assumption of this work is that an application designer has an complete and representative understanding of the situation to be detected, and thus is the appropriate person to construct a concept expression. We made this assumption, first, because we felt application designers would be equipped to express situations that are commonly-understood as socially-appropriate for an experience (e.g., a park used as an event venue is generally less appropriate for play activities like tossing a frisbee) or where characteristics of the physical environment would generally afford certain actions or activities (e.g., an open area supports tossing a frisbee); second, we believed that designers using Affinder would encode multiple, diverse concepts of a situation due to the nature of their task—which was to express a diverse set of contexts in which to engage in a digitally-mediated, shared experience—in order to promote more opportunities for socially connecting.

Nonetheless, issues can arise when an application designer's concept of situations is not commonly shared with the end-user who will encounter these situations, perhaps due to differences in personal preferences or cultural understanding. For example, end-users may have their own personal interpretation of a concept for a situation which differs from the application designer's concept of it (e.g., an end user might prefer soupy foods over spicy foods for "food good for a cold day"). If a context-aware application uses a concept expression which does not align with end-users' concepts of the situation, the app will try to facilitate the activity in situations that some end-users will not agree with. Therefore, a promising direction for future work is providing methods and tools that can account for multiple personal or cultural interpretations of the situations that are appropriate for activities. Future research might develop expression tools that would support multiple authors' involvement in the construction and customization of how concept expressions operate, in order to be accountable to these diverse end-users concepts of a situation. Pursuing such a direction would build upon foundational work on intelligibility, accountability, and control for end-user interaction with context-aware applications [4, 12]. Affinder's concept expressions—logical predicates which are composed of semantically-meaningful intermediate concepts and context-features—are in an intelligible representation that makes them good candidates for supporting this desired control and customization by other authors.

10 CONCLUSION

In this paper, we sought to enable designers of context-aware applications to more easily express their ideas of a conceptually-rich human situation and the interactions they afford to machines, so that the applications can be aware and responsive to such situations across distributed contexts. To do this, we created Affinder, a block-based programming environment that supports designers in following an effective construction for bridging from their human concepts of a situation into a machine representation using available context-features. Affinder's technical contribution includes 3 core features designed to overcome challenges when constructing concept expressions: (1) an *unlimited vocabulary search* for discovering features they may have forgotten; (2) *prompts for reflecting and expanding* their concepts used for organizing and foraging for features; and (3) *simulation and repair tools* for identifying and resolving issues with the precision of concept expressions on real use-cases. In our studies, we show that these features can (1) mitigate underscoped expressions by helping designers discover context-features relevant to their concepts, (2) stretch people's concepts for what aspects they consider to be important for enabling interactions in these situations, and (3) uncover and resolve mismatches in how a creator expected their concept expression to operates vs. how it actually executes across real-world, distributed contexts. Our work with Affinder represents an exciting direction for the development of intelligent and context-aware applications, where we explicitly focus on advancing the capabilities of humans to bridging between their ideas and available machine representations, rather than only work to develop better component detectors.

Our current design of Affinder provides numerous interface techniques to support effective cognition processes as designers bridge between their concepts and the machine representations. In future work, we envision that AI techniques like semantic embeddings and knowledge graphs could play a larger role in supporting the human during the construction process. As one example, using language embeddings [33] could enable ways to recommend concepts and vocabulary that would be better aligned with what humans actually want (e.g., if the term 'spacious' is returning undesirable place contexts that are indoors, we might add the term 'outdoors' to the embedding to direct the search). In another direction, commonsense knowledge graphs [5] could help users traverse a graph of related concepts to explore and discover new concepts.

ACKNOWLEDGMENTS

The authors thank Kapil Garg, Garrett Hedman, Spencer Carlson, Dan Rees Lewis, Kristine Lu, Katie Cunningham, Leesha Maliakal Shah, Yongsung Kim, and many other members of the Delta Lab at Northwestern for design feedback and literature help; the students of the DTR program at Northwestern for design and implementation feedback; and Carrie Cai, Andrés Monroy-Hernandez, Jeremy Birnholtz for their thoughtful insights on our contributions. This research was supported in part by a Google Faculty Research Award and by the National Science Foundation under award IIS-1618096.

REFERENCES

[1] Gregory D Abowd and Elizabeth D Mynatt. 2000. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction (TOCHI)* 7, 1 (2000), 29–58.
[2] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *Ai Magazine* 35, 4 (2014), 105–120.
[3] Linden J Ball and Bo T Christensen. 2019. Advancing an understanding of design cognition and design metacognition: Progress and prospects. *Design Studies* 65 (2019), 35–59.
[4] Victoria Bellotti and Keith Edwards. 2001. Intelligibility and accountability: human considerations in context-aware systems. *Human–Computer Interaction* 16, 2-4 (2001), 193–212.
[5] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia-A crystallization point for the Web of Data. *Journal of web semantics* 7, 3 (2009), 154–165.
[6] Michael Brooks, Saleema Amershi, Bongshin Lee, Steven M. Drucker, Ashish Kapoor, and Patrice Simard. 2015. FeatureInsight: Visual support for error-driven feature ideation in text classification. In *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*. 105–112. https://doi.org/10.1109/VAST.2015.7347637

[7] Nathan Crilly and Carlos Cardoso. 2017. Where next for research on fixation, inspiration and creativity in design? *Design Studies* 50 (2017), 1–38.

[8] David Dearman, Timothy Sohn, and Khai N. Truong. 2011. Opportunities Exist: Continuous Discovery of Places to Perform Activities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2429–2438. https://doi.org/10.1145/1978942.1979297

[9] David Dearman and Khai N Truong. 2010. Identifying the activities supported by locations with community-authored content. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*. 23–32.

[10] Anind K Dey, Gregory D Abowd, and Daniel Salber. 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human–Computer Interaction* 16, 2-4 (2001), 97–166.

[11] Anind K Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. 2004. a CAPpella: programming by demonstration of context-aware applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 33–40.

[12] Anind K. Dey and Alan Newberger. 2009. Support for Context-Aware Intelligibility and Control. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 859–868. https://doi.org/10.1145/1518701.1518832

[13] Anind K Dey, Timothy Sohn, Sara Streng, and Justin Kodama. 2006. iCAP: Interactive prototyping of context-aware applications. In *International Conference on Pervasive Computing*. Springer, 254–271.

[14] Paul Dourish. 2001. Seeking a foundation for context-aware computing. *Human–Computer Interaction* 16, 2-4 (2001), 229–241.

[15] Paul Dourish. 2006. Re-space-ing place: " place" and" space" ten years on. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. 299–308.

[16] Neil Fraser. 2015. Ten things we've learned from Blockly. In *Blocks and Beyond Workshop (Blocks and Beyond), 2015 IEEE*. IEEE, 49–50.

[17] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM* 30, 11 (1987), 964–971.

[18] Harish Haresamudram, Apoorva Beedu, Varun Agrawal, Patrick L Grady, Irfan Essa, Judy Hoffman, and Thomas Plötz. 2020. Masked reconstruction based self-supervision for human activity recognition. In *Proceedings of the 2020 International Symposium on Wearable Computers*. 45–49.

[19] Steve Harrison and Paul Dourish. 1996. Re-place-ing space: the roles of place and space in collaborative systems. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*. 67–76.

[20] Barbara Hayes-Roth and Frederick Hayes-Roth. 1979. A cognitive model of planning. *Cognitive science* 3, 4 (1979), 275–310.

[21] Ilyena Hirskyj-Douglas, Anna Kantosalo, Andrés Monroy-Hernández, Joelle Zimmermann, Michael Nebeling, and Mar Gonzalez-Franco. 2020. Social AR: Reimagining and Interrogating the Role of Augmented Reality in Face to Face Social Interactions. In *Conference Companion Publication of the 2020 on Computer Supported Cooperative Work and Social Computing*. 457–465.

[22] Scott Hudson, James Fogarty, Christopher Atkeson, Daniel Avrahami, Jodi Forlizzi, Sara Kiesler, Johnny Lee, and Jie Yang. 2003. Predicting human interruptibility with sensors: a Wizard of Oz feasibility study. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 257–264.

[23] Yelp Inc. 2020. Yelp Open Dataset. https://www.yelp.com/dataset.

[24] Yelp Inc. 2021. Yelp Fusion API. https://www.yelp.com/developers/documentation/v3.

[25] Tanya R Jonker, Ruta Desai, Kevin Carlberg, James Hillis, Sean Keller, and Hrvoje Benko. 2020. The Role of AI in Mixed and Augmented Reality Interactions. In *CHI2020 ai4hci Workshop Proceedings. ACM.*

[26] Harmanpreet Kaur, Alex C Williams, Daniel McDuff, Mary Czerwinski, Jaime Teevan, and Shamsi T Iqbal. 2020. Optimizing for happiness and productivity: Modeling opportune moments for transitions and breaks at work. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–15.

[27] Gierad Laput and Chris Harrison. 2019. Sensing fine-grained hand activity with smartwatches. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[28] Julie S. Linsey, Arthur B. Markman, and Kristin L. Wood. 2012. Design by Analogy: A Study of the WordTree Method for Problem Re-Representation. *Journal of Mechanical Design* 134, 4 (04 2012). https://doi.org/10.1115/1.4006145 arXiv:https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/134/4/041009/5606684/041009_1.pdf 041009.

[29] Google LLC. 2021. Google Awareness API. https://developers.google.com/awareness/.

[30] Ryan Louie, Andy Coenen, Cheng Zhi Huang, Michael Terry, and Carrie J Cai. 2020. Novice-AI music co-creation via AI-steering tools for deep generative models. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[31] Ryan Louie, Jesse Engel, and Anna Huang. 2021. Expressive Communication: A Common Framework for Evaluating Developments in Generative Models and Steering Interfaces. arXiv:2111.14951 [cs.HC] https://arxiv.org/abs/2111.14951

[32] Ryan Louie, Kapil Garg, Jennie Werner, Allison Sun, Darren Gergle, and Haoqi Zhang. 2021. Opportunistic Collective Experiences: Identifying Shared Situations and Structuring Shared Activities at Distance. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW3 (2021), 1–32.

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546* (2013).

[34] Mehrab Bin Morshed, Koustuv Saha, Richard Li, Sidney K D'Mello, Munmun De Choudhury, Gregory D Abowd, and Thomas Plötz. 2019. Prediction of mood instability with passive sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 3 (2019), 1–21.

[35] Juan Ramos. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the First instructional Conference on machine learning*, Vol. 242. 133–142.

[36] Alejandro Rivero-Rodriguez, Paolo Pileggi, and Ossi Antero Nykänen. 2016. Mobile Context-Aware Systems: Technologies, Resources and Applications. *International Journal of Interactive Mobile Technologies (iJIM)* 10, 2 (Apr. 2016), pp. 25–32. https://doi.org/10.3991/ijim.v10i2.5367

[37] Daniel Salber, Anind K Dey, and Gregory D Abowd. 1999. The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 434–441.

[38] Pao Siangliulue, Kenneth C. Arnold, Krzysztof Z. Gajos, and Steven P. Dow. 2015. Toward Collaborative Ideation at Scale: Leveraging Ideas from Others to Generate More Creative and Diverse Ideas. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Vancouver, BC, Canada) *(CSCW '15)*. Association for Computing Machinery, New York, NY, USA, 937–945. https://doi.org/10.1145/2675133.2675239

[39] Pao Siangliulue, Joel Chan, Steven P. Dow, and Krzysztof Z. Gajos. 2016. IdeaHound: Improving Large-Scale Collaborative Ideation with Crowd-Powered Real-Time Semantic Modeling. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 609–624. https://doi.org/10.1145/2984511.2984578

[40] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L Littman. 2014. Practical trigger-action programming in the smart home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 803–812.

[41] Xiaotong (Tone) Xu, Rosaleen Xiong, Boyang Wang, David Min, and Steven P. Dow. 2021. IdeateRelate: An Examples Gallery That Helps Creators Explore Ideas in Relation to Their Own. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW2, Article 352 (oct 2021), 18 pages. https://doi.org/10.1145/3479496