Stakeholder Preference Extraction from Scenarios

Yuchen Shen*, Travis Breaux†
Software and Societal Systems Department, Carnegie Mellon University
Email: *yuchens2@andrew.cmu.edu, †breaux@cs.cmu.edu

***Abstract*—Companies use personalization to tailor user experiences. Personalization appears in search engines and online stores, which include salutations and statistically learned correlations over search-, browsing- and purchase-histories. However, users have a wider variety of substantive, domain-specific preferences that affect their choices when they use directory services, and these have largely been overlooked or ignored. The contributions of this paper include: (1) a grounded theory describing how stakeholder preferences are expressed in text scenarios; (2) an app feature survey to assess whether elicited preferences represent missing requirements in existing systems; (3) an evaluation of three classifiers to label preference words in scenarios; and (4) a linker to build preference phrases by linking labeled preference words to each other based on word position. In this study, the authors analyzed 217 elicited directory service scenarios across 12 domain categories to yield a total of 7,661 stakeholder preferences labels. The app survey yielded 43 stakeholder preferences that were missed on average 49.7% by 15 directory service websites studied. The BERT-based transformer showed the best average overall 81.1% precision, 84.4% recall and 82.6% F1-score when tested on unseen domains. Finally, the preference linker correctly links preference phrases with 90.1% accuracy. Given these results, we believe directory service developers can use this approach to automatically identify user preferences to improve service designs.**

***Index Terms*—requirements engineering, elicitation, stakeholder preferences, natural language processing**

I. Introduction

Software personalization concerns the tailoring of software features to better meet individual user desires. This includes *static personalization*, in which designers study narrow segments of demographics in order to identify personalization requirements and introduce options that satisfy those requirements at design-time, and *dynamic personalization*, which includes building user models to personalize features at runtime [19]. In this paper, we study techniques to improve static personalization by focusing on *directory services*, which describe services that support users who are searching for a product, service or experience. Directory service design spaces offer a rich, diversified and highly personal source of user preferences, and software that provides these services are becoming increasingly integrated into user experiences. Moreover, a variety of directory services exist today, from apartment and restaurant finding, to parking location and gas station finding, to discovering hiking trails and new music.

Directory services that use static personalization rely on large databases of information that have been organized around a generally closed set of user preferences. The OpenTable web and mobile application (app) is one such example, which allows users to search for and reserve dining tables at popular restaurants. In 2019, OpenTable served over 60,000 restaurants and over 1.6 billion diners worldwide [51]. In contrast, in 2021 the AllTrails.com web app serves a specialized hiking community of over 25 million registered users by indexing over 200,000 trail guides across 190 different countries [55]. Directory service apps often integrate with third-party services and allow users to rate their experiences of services and products found through the apps to improve the quality of data stored by these apps.

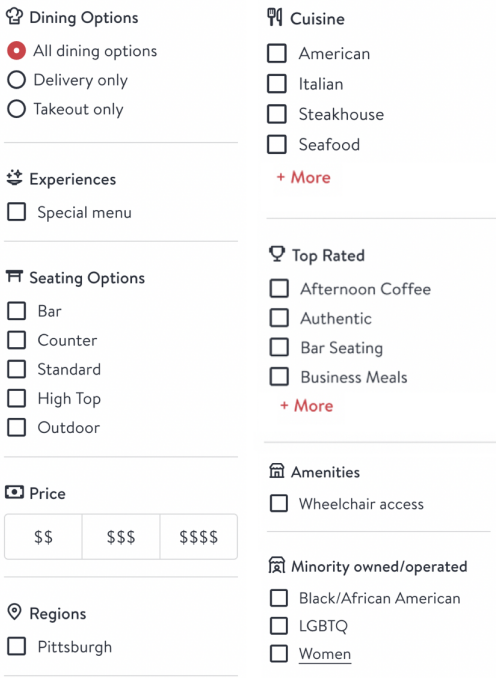

Figure 1. Example Preferences in OpenTable App

Figure 1 shows a set of preferences presented by the web app OpenTable that users can use to search: in addition to general preferences for dining option, price and region, the app also allows users to filter by a set of 46 narrower "Top Rated" preferences, including restaurants good for Afternoon Coffee, Fit for Foodies, Gluten Free, Quiet Conversation and Scenic View, to name a few. These preferences represent how users experience the service they are seeking and have become personalization requirements satisfied by filters in the OpenTable app. As designers of directory services increase their knowledge of user preferences, they can further tailor these services to increase personalization and user satisfaction.

Generally, preferences describe constraints on what users

want. This could include an *affordable* apartment, in a *safe* neighborhood, or a music library that *offers classical music*, or whether or not a hiking trail is *long* or *short*. Preferences are non-functional requirements, because they express a stakeholder's desire for a quality or attribute monitored by a system [23]. In Jackson's view of requirements engineering [29], this system includes the environment about which requirements are expressed, and software may have limited visibility into the state of that environment. As software increasingly integrates data to maintain environmental models, this software will be better situated to satisfy a broad range of stakeholder preferences. Today in directory services, these qualities are observed through data collected when users complete online surveys to rate their experiences of the results found through the directory. In the future, we believe opportunities will increase to enhance how these services monitor stakeholder preferences. For example, public crime statistics can be used to assess if a neighborhood is safe, category labels on music albums can be used to determine if music is classical, and wearable fitness data could be used to measure the distance and difficulty of a hiking trail. Richer environmental models that better distinguish stakeholder preferences can also improve how users find what they desire. While preferences have broad applicability across many types of systems, in this paper we focus specifically on *directory services*, which users query to find a thing of interest. This includes *unified services* that index a catalogue of interesting things using prepared categories of user preference, such as an apartment listing, a job listing or a hiking trail-finding application. Alternatively, users may rely on a *collection of unaffiliated services* to find a thing of interest. For example, a user interested in finding an academic degree program, may combine information from general search services, university websites, university rankings, job listing websites, and blogs, to differentiate degree programs. In both situations, users have preferences about what they are looking for, and are using services with varying levels of personalization to aid in discovering their things of interest. In this paper, we are primarily studying unified services and how to enrich their environmental models to provide greater personalization to users.

In agile software development, developers are frequently experiencing pressure to deliver increments of working systems that broadly satisfy a wide variety of users. This approach can overlook the more nuanced stakeholder preferences that lead to improved personalization. Because agile development has led to limited documentation (e.g., replacing use cases with user stories), we recognize that developers need automated methods to extract requirements to meet the demands of personalization. In this respect, we propose to elicit user stories directly from users, and deliver to developers those preferences that stakeholders describe without the effort to manually analyze those scenarios.

To that end, we describe research to extract user preferences from user-authored scenarios. The contributions of the research include (1) a grounded theory based on the analysis of 217 unique user scenarios elicited over 12 different directory service domains. The grounded theory describes the collective phenomena that comprise stakeholder preferences in these domains, including heuristics to identify and code these phenomena, resulting in 7,661 preference annotations. (2) A survey of 15 popular directory service websites to identify gaps between what stakeholders desire (stakeholder preferences) and what those services offer (app features). (3) Three supervised machine-learning models to automatically extract preferences from scenarios based on three different approaches: a classical model using conditional random fields over word distributions; a random forest trained on typed dependencies; and a transformer trained on BERT word embeddings. Finally, (4) we present a static technique to link preference labels in a scenario into a preference phrase that describes a preference requirement for use in the design of a web or mobile app. Finally, we illustrate how the extraction and linking tools can be used in an end-to-end preference elicitation pipeline. This work is novel because, to our knowledge, it's the first end-to-end tool to extract stakeholder preferences from scenarios. Our dataset and supporting tools, including a trained transformer model that developers may use, are available online [73].

This paper is organized as follows: in Section 2, we present background and related work; in Section 3 we present our approach, including studies to construct the grounded theory, survey existing systems, and extract and link preferences; in Section 4, we report our evaluation and results; in Section 5, we present our discussion; and in Section 6, we discuss threats to validity, and we conclude in Section 7.

II. Background and Related Work

We now review background and related work on preferences, requirements extraction, and software personalization.

A. Goals, Desires and Preferences

Personalization has a history of motivation in one-to-one marketing, in which companies track users and promote their products and services to individual user interests [54]. Automated techniques for one-to-one marketing rely on user modeling [19], in which companies track: user-to-item affinities, such as search queries and product views; item-to-item affinities, such as correlations among product views by the same user; and user-to-user affinities, including classifying users by similar actions and item interests [61]. These three affinities are relatively easy to measure from user telemetry data, and thus have underpinned automated recommender systems over the past two decades. One disadvantage of relying on these affinities, however, is that user needs are expressed as correlations drawn from non-observable, latent variables that may explain why users take actions or prefer specific items, without indicating what characteristics of products and services users actually prefer.

In requirements engineering, stakeholders are people who have a stake in the system, including project managers, customers, and users [69]. Stakeholder requirements may be described as *desirable*, meaning they would be nice to have, or *necessary*, meaning the system cannot be built without

satisfying those requirements. Requirements can be expressed as *high- and low-level goals* to be achieved or maintained by the system [15], and this includes system qualities or *soft-goals* [48], such as privacy, safety, etc. Desirable and necessary goals have been called optional and mandatory, respectively [39]. In this case, a desirable (or optional) goal is one that a developer may choose to implement or ignore, because it is necessary. Similarly, preferences, which include stakeholder desires, can be understood by developers to be optional. However, we believe preferences can be indicators of opportunities to innovate and better meet stakeholder needs.

Soft-goals, which describe both qualities of the system, such as performance, and qualities of the environment and stakeholder experience, such as privacy, intersect with *human values*, which include larger societal concerns, such as social justice and transparency [45]. Stakeholder preferences are expressed as priorities over high-level qualities, or as desired operational details [39]. Unlike necessities, all stakeholder preferences need not be satisfied. Stakeholder preferences may include pro-social qualities as well as pro-ego qualities that benefit the individual over the group. Maximizing stakeholder preferences satisfaction supports greater software personalization, as software features are tailored to the individualized needs of smaller, similar stakeholders groups.

Requirements elicitation has been the subject of significant study [20], [36], [52]. Requirements can be acquired using various techniques, such as: analyzing the direct and indirect system stakeholders to determine stakeholder needs and generate user stories [17]; combining object-oriented system analysis [25], [49], task analysis and prototyping to better account for end-users and the usage context [64]; gathering informal text descriptions of user requests with crowd sourcing methods, and automatically classifying those requests into requirements classes based on keywords [38], among others. Techniques can also be used to enhance creativity during the elicitation of user stories [46]. The tasks of distinguishing between mandatory and optional stakeholder goals and building tools that efficiently search for alternatives to best satisfy a given set of mandatory and preferred requirements has also been studied [39], [40].

B. Requirements Extraction

Techniques to extract requirements from texts have been used to identify legal primitives in laws [32], mine software requirements in app reviews [31], [66] and to perform domain modeling [3]. Legal compliance checking can be automated through the extraction of compliance requirements from legal documents [32]. Jha and Mahmoud describe a two-stage process for mining non-functional requirements from mobile app reviews, noting that 40% of app reviews include at least one non-functional requirement [31]. Domain models that describe knowledge about domains [8] can be built using information retrieval and natural language processing techniques for automatically extracting model elements from text using rules based on natural language dependencies [3].

Typed dependencies correspond to syntactic and grammatical relationships between words in natural language (NL), such as the nominal subject of a sentence, or the direct object of a verb [43]. Dependencies have been used in requirements engineering to extract assertions from NL specifications for use in formal verification [63], legal meta-data from regulations [62], and software features from user manuals [57]. They have also been used to classify requirements as either functional or non-functional [14], [33], and to demarcate requirements in a specification [2]. Conceptual models have been extracted from user stories using syntactic rules that resemble typed dependencies (e.g., nsubj and nmod) [58]. Natural language syntax varies widely, even when describing the same concept: e.g., the two sentences "the large apartment has two bedrooms" and "the apartment, which has two bedrooms, is large" yield two different dependency graphs. Thus, successful typed dependency usage require analyzing large corpora, or a narrow domain with few syntactic variations used to express requirements. While current work using typed dependencies with machine learning classification shows promise [2], [14], [33], more work is needed to generalize these approaches.

Another method that has been used to extract requirements is Named Entity Recognition (NER) [21], [26]. Named Entity Recognition is an information extraction method that recognizes word sequences in unstructured text, and classifies them into predefined categories, called named entities, such as people's names, organizations, dates, and locations. Dictionary-based NER techniques use gazetteers, i.e., lists of as many predefined named entities as possible, since recognition and classification performance relies on the completeness of the dictionary [65]. The approach is simple but limited when recognizing unseen words. Rule-based techniques use rules or patterns drawn from real sentences where named entities occur [9]. This approach addresses the limitation of unseen words, but is limited because a wide variety of texts are hard to summarize in a complete set of rules, and all rules must be written prior to extraction. Machine learning (ML) techniques include support vector machines (SVMs) [12], conditioned random fields (CRFs) [41], and neural network models, such as bidirectional long-short term memory (Bi-LSTM) [27], convolutional neural nets (CNNs) [11], and transformers [68]. These methods do not rely on predefined rules, rather the models are trained to learn statistical patterns from data. The highest F1 score achieved using machine learning on the CoNLL 2003 named entity recognition dataset is 94.3% [67]. A limitation is the need for large training corpora, which is being addressed by emerging unsupervised and semi-supervised ML techniques [50].

In our work, we wish to extract requirement-related information from text corpus. To do this, we apply three ML models. First, the CRF model is a graph-based model that learns patterns in the words that appear before and after a named entity, and computes the bi-directional flow of probabilistic information across word sequences. We choose the CRF model as a baseline to our task, because it was the state-of-the-art prior to neural models [30]. Second, the Random

Forest model learns multiple decision trees using randomly selected subsets of the data chosen to optimize performance. The Random Forest is trained on the "neighborhood" of typed dependencies before and after a named entity. We choose typed dependencies as a scenario text representation to complement word distributions, and because they are popular in automated requirements analysis [14]. Third, the Bidirectional Encoder Representations from Transformers (BERT) model [13] is a transformer-based language representation model that utilizes the attention mechanism to learn both relations between tokens and relations between sentences in a text. We choose BERT because it obtains state-of-the-art results on various natural language processing tasks, including named entity recognition [67].

C. Software Personalization

Personalization began in product manufacturing in the 1980's, when Stanley Davis [16] first argued to shift manufacturing from mass production to mass customization, in which companies employ agile processes and delayed differentiation to tailor products to individual customer needs [35]. In the last two decades, the shift toward personalization in information systems has outpaced manufacturing, as software developers combine agile methods [5], A/B or split testing [34], real-time user telemetry, and automatic updates to rapidly optimize software and increase user dwell time and engagement. To date, personalization is largely based on statistical models of user-system interaction, with little documentation of what users prefer when they evaluate information online by using directory service applications.

D. Comparison with Other Preference Elicitation Approaches

Our preference-based elicitation method complements traditional elicitation approaches, such as personas [22], app reviews and issues [47]. Compared to these approaches, our approach situates stakeholder needs in the context of a rich description that is technology-independent. Developers can then envision how users would use their system in a real-world setting, including the goals that users wish to achieve and potential obstacles they would meet. Personas provide developers with imaginary users from which they can envision requirements. In our approach, we collect requirements from real stakeholders and prospective users. In app reviews, the requirements are entailed by the implementation and users will only request new features when describing missing requirements. Our approach is not bound to a specific implementation, and can yield user needs without a presumption of how those needs would be realized by specific features. This provides developers more flexibility in how to translate needs into requirements and features. Finally, issues are similar to app reviews and are also bound to specific requirements, whether they be errors or feature requests.

Moreover, our scenario-based preference elicitation method elicits and documents preference from potential directory service application users, contradictory to traditional personalization methods that are largely based on statistical models of user-system interaction. We believe our method allows developers to obtain a more personalized understanding of the users, which in turn further improves software personalization.

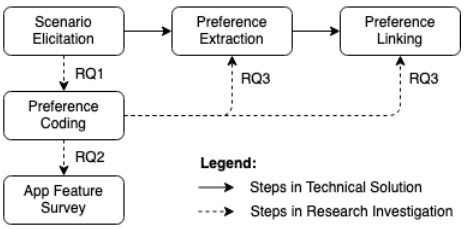

Figure 2. Research Method Overview

We conducted an unpublished survey to directly elicit preferences from stakeholders, and discovered that direct elicitation requires first naming all the entities of interest (e.g., a good neighborhood is [fill in blank]), whereas scenario-based elicitation permits users to explore and propose entities they prefer, especially those related to but not contained in the scenario prompt. Hence we opted to use the scenario-based elicitation approach.

III. Approach

Our research aims to answer the following research questions (RQs):

RQ1: How do stakeholders describe their individual preferences in scenarios they author?

RQ2: To what extent are the elicited stakeholder preferences satisfied by existing directory services?

RQ3: What natural language features are best suited to automatically extract preferences from scenarios?

The approach consists of the following steps (see Figure 2), which map onto the above RQs: (1) eliciting preference-laden scenarios from stakeholders; (2) coding of the scenarios by the researchers to discover a grounded theory of stakeholder preferences; (3) surveying the features of top web-based directory service applications to evaluate whether or not they satisfy elicited stakeholder preferences; (4) training three classifiers to automatically extract preference words from scenarios using conditional random fields (CRF), random forest (RF), and bidirectional encoder representations from transformers (BERT); and (5) designing a static technique to link extracted words into preference phrases. In Figure 2, the research steps connected by dotted lines are used to develop a solution to automate preference extraction, connected by solid lines. In this figure, RQ1 is answered by eliciting scenarios from users and coding the preferences within the user-authored scenarios to describe how stakeholders express their preferences. RQ2 is answered by summarizing the coded preferences and conducting an app feature survey to identify gaps in which apps do not satisfy user preferences. Finally, RQ3 is answered by using the coded preferences to train three natural language models and to predict links between preference elements to yield requirements.

We now describe the technical details of each step and how they relate to the research questions in the approach, below.

A. Scenario Elicitation

The scenario elicitation study is designed to invite authors to answer a single question prompt by writing a minimum of 150 words. The prompts were created to reflect a variety of directory services, from unified services, where a web applications (apps) assist users in answering the question, to unaffiliated services, where users forage for their answer by combining information from multiple web apps. For instance, a user may look at a single apartment-finding website to find a desired apartment, or may choose to forage through several websites in order to choose a school or degree problem, likely including school ranking websites, schools' official websites, and websites that contain reviews to each school. This study extends a previous study conducted by Shen et. al [60] that collected scenarios for 4 prompts with additional scenarios in response to 8 new prompts collected for this study. This yields a total of 139 new scenarios beyond the original 78 scenarios collected by Shen et. al [60]. The prompts used in this step appear in Table I. Each prompt asks the author to describe the steps they take to answer the question, and does not require the author to use apps to complete the steps. In addition, authors may have varying degrees of experience with and knowledge about each activity queried. Thus, the elicited scenarios will include incompleteness, vagueness and ambiguity, which is expected with a single-prompt elicitation exercise. We use the *category name* of a prompt to refer to the scenarios elicited using that prompt.

Table I
SCENARIO PROMPTS LISTED BY CATEGORY

Category	Scenario Question Prompt
Apartment	How do you find an apartment?
Restaurant	How do you choose a restaurant to eat at?
Hiking	How do you plan a trail hike in a park?
Health Clinic	How do you choose a clinic to visit when you get sick?
Flight Booking	How do you book an airline ticket?
Social Net	How do you stay connected with friends?
Movies	How do you choose a movie to watch?
Hotel Booking	How do you choose and reserve a hotel room?
Degree Programs	How do you choose a school or degree program?
Job Hunting	How do you search for a new job?
Travel	How do you choose a place to travel to?
Course Selection	How do you choose an elective course for enrollment?

The scenarios were collected using Amazon Mechanical Turk in an IRB-approved research study. Workers volunteered and consented to participate by accepting the human intelligence task (HIT), and were then provided the question, e.g., "How do you choose a clinic to visit when you get sick?" and asked to write a scenario that includes four elements: *steps* in the process; *goals* that the author wants to achieve; *preferences* or qualities the stakeholder pays attention to during the process of trying to achieve the goals; and *obstacles* that could go wrong with the process, and how the stakeholder reacts. We asked the authors to include goals, steps and obstacles in their scenarios in addition to preferences, to encourage authors to identify additional context, consisting of nouns and verbs, that trigger thoughts about a wider range of preferences. When asked about how to choose a restaurant to eat at, for example, in addition to mentioning food quality and restaurant environment as two example preferences, an author should include the *steps* taken, such as going online to read reviews, the *goals* the author wants to achieve, such as staying in a comfortable and quiet environment, and the *obstacles* the author may face, such as being unable to get the ordered food in a timely manner.

Eligible workers completed over 5,000 HITs, have an approval rating greater than 97%, and are located in the United States. Scenarios were rejected if they did not respond to the prompt, if they did not contain the required four elements, or if they contained more than five spelling, capitalization or grammar errors. For example, some authors completed the task by copy pasting irrelevant text from the internet, or by just writing one or two irrelevant sentences that failed to contain the four elements. Workers were paid $2.00 for each accepted scenario, and workers who completed a scenario in the top 20% of ranked scenarios were paid an additional bonus of $1.00. For deciding which scenarios should receive the bonus, scenarios were ranked based on their coverage of the four elements. Both authors manually inspected the scenarios and agreed on the ranking. Finally, worker identifiers were removed from the scenarios prior to analysis.

B. Preference Coding

Grounded theories are "grounded" in the data, which can yield strong evidence to support the theory, but which also limits generalizability [10]. In this paper, we discover descriptive theory of how stakeholders express preferences in text, which is characterized by a typology to recognize those words and combine those words into preference phrases in a reliable and repeatable way. We use qualitative coding of scenarios in multiple cycles to identify these words and phrases [59].

To answer RQ1, how do stakeholders describe their individual preferences in scenarios they author, the authors coded the scenarios using a three-cycle coding process to produce the grounded theory [59]. The annotations were applied using a simple markup, which was parsed using regular expressions. In the first cycle, the authors used initial or "open coding" [59] to code all phrases that express what stakeholders prefer when answering the scenario question using a sample of seven scenarios drawn from four domains (Apartments, Restaurants, Hiking, and Health). Next, they reviewed each coded phrase and separated the phrase into disjoint sub-codes in a second cycle, called axial coding, which was applied to the same seven scenarios. Saturation was achieved in the second-cycle when no new sub-codes were identified. The resulting closed coding frame that comprises the grounded theory [59] is as follows:

- entity - a noun phrase about which a preference can be stated, including any preceding adjectives, .e.g., “good meal,” “best place,” or “price”
- modifier - an adjectival clause that modifies an entity and indicates a preference but is not directly before an entity, e.g., “good”, “quiet”
- relation - a prepositional word that is attached to an entity and indicates a valued relationship between two entities, e.g., “along” in “a swimming area along the hike”
- action - is a verb phrase that indicates either (1) a distinguishing value of an entity, e.g., “teaches” in “a destination that teaches its culture to me”; or (2) a stakeholder goal that does not describe a step in the scenario, e.g., “waste” in “waste a lot of gas”

While the coding frame is generic, it’s application is limited to words and phrases about which stakeholders have expressed preferences. Thus, not all noun phrases are coded as entities, nor are all adjectives coded as modifiers. Techniques such as part-of-speech tagging, typed dependencies and semantic role labeling alone cannot predict these phrases without crafting numerous individual rules that are unlikely to generalize. Instead, training a classifier aims to predict which noun phrases and adjectives correspond to the codes for preferences.

Finally, both authors separately performed a third cycle using theoretical or “closed coding” [59] by applying the coding frame to relevant words in a new sample of 20 scenarios. Heuristics were developed independently to aid the coders in recognizing when to include or exclude words from annotations in the third cycle. The final heuristics used were: annotations cannot overlap, determiners are excluded (“a,” “an,” “the”), and avoid splitting conjunctions when adjectives are present (“safe area and neighborhood”). After independently coding the 20 scenarios, the authors measured inter-rater reliability to yield 0.67 Cohen’s Kappa, which Landis and Koch report as “substantial” agreement [37]. After reviewing and resolving differences in the sample, updating the coding heuristics, and re-coding the same sample, both authors achieved a 0.94 Kappa, which shows high above-chance agreement acceptable in computational linguistics [1].

Figure 3 shows an example sentence with each code represented from the coding frame. This example is unusual, because it represents all four codes; sentences in the corpus average 2.7 entities and 0.3 modifiers per sentence. Summary statistics on the distributions of preference types across scenarios appear in the results in Table III.

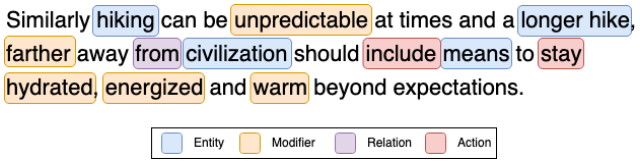

Figure 3. Example Sentence Coded with Coding Frame

C. App Feature Survey

Preferences clarify what kinds of entities stakeholders prefer when they are engaging in an activity. With regard to directory services, if a stakeholder prefers a “pet-friendly” apartment, or a hiking trail “with elevation gain,” then a robust service could provide functionality to discern which entities satisfy their preferences. To answer RQ2, to what extent are the elicited stakeholder preferences satisfied by existing directory services, and as a further motivation for the importance of our preference extraction tool, we identified popular directory service apps for a selection of scenario prompts and checked whether the applications allow stakeholders to express their preferences. Popular apps were identified from the top twenty results of a Google search using keywords from four scenario questions for apartments, restaurants, hiking, and health clinics. Next, the labeled preferences elicited from stakeholders were grouped by the second author into feature categories that cover the preferences. The categories were derived by comparing labeled preferences for similar meanings (e.g., the “distance to work” category includes preferences labeled “close to my work,” “close to where I work,” etc.) An app satisfies a stakeholder preference if it contains a software feature that matches the category associated with that preference. Evidence for a software feature shown in a web app include search filters, check boxes, fields in a result table, and tags or badges, among others. For example, in Table II in the first column, we present a non-exhaustive list of examples of labeled phrases in brackets, followed by the assigned label in the second column, and assigned feature category in the third column. Note that we only put brackets around phrases with the specific label stated in the Label column, and omitted other brackets in the phrase for better clarity. For example, for the labeled phrase “distance [to] work”, we omitted brackets around the entities [distance] and [work]. Brackets will be used throughout this paper to indicate labeled words and phrases.

Table II
EXAMPLE PREFERENCES WITH FEATURE CATEGORIES

Labeled Phrase	Label	Feature Category
[afford]	action	Affordable
too [costly]	modifier	Affordable
distance [to] work	relation	Commute
[in] walking distance	relation	Commute
[studio] or [one bedroom]	entity	Layout
[upstairs unit]	entity	Layout
[disable friendly]	modifier	Accessible
[wheelchair]	entity	Accessible
[feel] safe	action	Safety
[less-safe neighborhood]	entity	Safety

When reviewing a web app, if the app includes a search filter to restrict price, then this feature would match the “affordable” feature category, because it could be used by a stakeholder to decide which apartments they can “afford,” or to filter out apartments that are “too costly.” The first and second authors coded the web apps using the feature categories [59], and the results are presented in Section IV-A.

D. Preference Extraction

We sought to answer RQ3, what natural language features are best suited to automatically extract preferences from sce-

narios, by investigating three approaches: a classic conditional random field trained on word distributions, a random forest trained on typed dependencies, and a transformer using BERT word embeddings. For all of these three approaches, we use the result from preference coding, as demonstrated in Section III-B, as the ground-truth training data. We explain how each of the three approaches are applied by us, and why we chose them, below. We present our evaluation of the three trained models in Section IV-B.

1) Rational for Model Choices: The first model we chose to use is the Conditional Random Fields (CRFs) model. We chose the CRF model as a baseline for our automated preference extraction task, because CRFs have long been a top performer in named entity recognition [53], and the Stanford Named Entity Recognizer provides a reliable implementation [18]. The second model we chose is a random forest classifier trained on typed dependencies. We chose typed dependencies as a scenario text representation to complement word distributions, and because they are popular in automated requirements analysis [14]. The last model we chose is a Bidirectional Encoder Representations from Transformers (BERT)-based language representation model. Due to BERT's strong performance in many language-related tasks, including named entity recognition [67], we choose to apply BERT to our task of preference extraction.

2) Conditional Random Fields: The Conditional Random Fields (CRFs) model is the baseline in our automated preference extraction task. The training files are prepared by first word-tokenizing the scenario text and labeling each token based on whether it is at the beginning (B), inside (I), outside (O), or at the end (E) of an entity label, called the BIOES representation. In this study, the entities correspond to one of the four codes (entity, modifier, relation, action) described in Section III-B. Figure 4 presents a training data example, where tokens (words, punctuation, etc.) are tab-separated from labels (O, B-ENTITY, etc.) and each line is numbered from 1 to 17 for presentation purposes. The labels include positional information encoded using the BIOES representation: when an entity *begins*, the position code "B" is prefixed to the entity category label; *inside* a multi-token entity, the position code "I" is prefixed; and the last token in a multi-token entity is prefixed with the *end* code "E"; otherwise, the code "O" is used to indicate tokens *outside* the labeled entity. In Figure 4, lines 6, 9 and 16 show a single-token named entities "clinic", "integrity" and "patient" for the ENTITY label. Line 14 show a single-token named entities "with" for the RELATION label. Lines 11-13 present multi-token named entities for the ENTITY label.

3) Typed Dependencies: Typed dependencies describe asymmetric, grammatical relationships between word pairs in a sentence, such as whether a word is the subject or direct object of a verb [43]. In Figure 5, we present a directed typed dependency graph for an elicited sentence expressing a stakeholder preference on a health clinic. The stakeholder expresses two preferences: the clinic has "integrity" and the clinic has "great communication skills." Knowing the "clinic" word position in

```
1    I              O
2    like           O
3    to             O
4    pick           O
5    a              O
6    clinic         B-ENTITY
7    that           O
8    has            O
9    integrity      B-ENTITY
10   and            O
11   great          B-ENTITY
12   communication  I-ENTITY
13   skills         E-ENTITY
14   with           B-RELATION
15   the            O
16   patient        E-ENTITY
17   .              O
```

Figure 4. Example Training Data for CRF-based Named Entity Recognizer

the graph, one can reach the first preference via two edges in the graph: the *acl:relcl* (relative clause modifier) and *obj* (direct object) relations. The second preference can be reached via a third edge, the *cc* (coordination) relation. We trained a

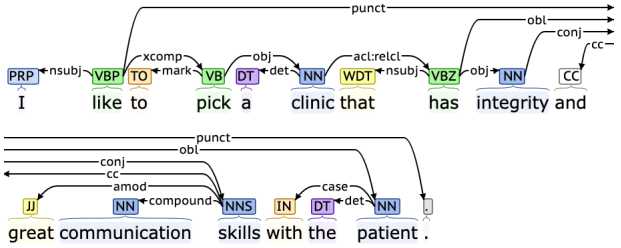

Figure 5. Example Typed Dependency Graph

random forest classifier using the typed dependencies acquired using the Stanza library [56]. Unlike the CRF and BERT-based transformer, which learn from word distributions and word embeddings, typed dependencies can be used to explain or rationalize the predictions of the random forest based on the grammatical relations between words.

The random forest was trained by vectorizing the coded scenarios as follows: each word vector of length $2w$ - 1 consists of the universal part-of-speech and dependency relation for up to w other words found in the graph traversal to and from the word represented by the vector. For example, the 3-word context for the word "clinic" in Figure 5 would yield two vectors[1]: [DT, det, NN, obj, VB] and [VBZ, acl:relcl, NN, obj, VB]. The first w-1 elements in each sequence follow dependency relations from governor to dependent, whereas the last w-1 elements follow relations from dependent to governor. If the traversal reaches an endpoint (e.g., the "root" of the graph, or word without a governor role), then the vector is zero-padded up to the length of the vector. Because a word can be the governor in multiple relations, the word can

[1]Figure 5 shows Treebank part-of-speech tags, which total 36 possible tags, whereas there are only 17 universal tags, e.g., the three adjectival tags JJ, JJS and JJR in Treebank map to one universal tag ADJ.

be described by multiple dependency paths or vectors. Each vector is finally labeled with the ground truth label, if the word is annotated in the ground truth, or with *no-label*, otherwise.

4) Bidirectional Encoder Representations from Transformers: The Bidirectional Encoder Representations from Transformers (BERT) [13] is a transformer-based language representation model that utilizes the attention mechanism in a recurrent neural network to learn contextual relations in text. BERT is pretrained with two unsupervised tasks, namely a masked language task and a next sentence prediction task, to understand relations between tokens and relations between sentences. The parameters of BERT are then fine-tuned to adapt to specific downstream tasks, in our case the named entity recognition task. We use a pretrained DistilBERT model [70], downloaded from the HuggingFace library [71], and fine-tuned the model for preference extraction. This DistilBERT model is a distilled version of BERT with 40% fewer parameters than BERT. DistilBERT reduces the size of the BERT model for faster training and inference, while still retaining 97% accuracy of the full BERT on multiple language tasks. We choose it because it is a smaller model and requires significantly less computational power to train. The transformer is trained using the same training files that we used to train the Conditioned Random Fields model. We used the train-validation-test split approach to obtain an unbiased, trained model estimate to select between final models. The validation error serves as the objective function for hyperparameter tuning.

E. Preference Linking

Preference extraction described in Section III-D yields words labeled with the types *modifier*, *relation*, and *action* that must be linked to an *entity* to form a preference phrase. Approaches to link such types based on part-of-speech include regular expressions, constituency parsing and typed dependency parsing [28]. In practice, these approaches are tailored to specific examples and can be hard to generalize due to the variability and ambiguity in natural language syntax. As a baseline solution, we introduce a simple static technique based on word position as follows: for a list E of n triples, each corresponding to the i-th labeled word in $0 \leq i \leq n$ and consisting of the labeled word w_i, word position p_i, and preference label l_i, let $E = (w_1, p_1, l_1), ..., (w_n, p_n, l_n)$, and for each non-entity labeled word w_i with word position p_i, find the nearest entity labeled word w_j with word position p_j, with the minimum absolute distance between word positions $|p_i - p_j|$. Figure 6 shows an example with entities highlighted in blue, and modifiers in orange: using this technique, the nearest entity to "studious" would be "school environment", which is four words apart, versus "campus life," which is five words apart.

We present evaluation and results for linking in Section IV-C

IV. Evaluation and Results

We now report the results of eliciting and analyzing scenarios for preferences. The scenario corpus was constructed by

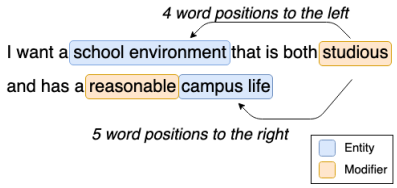

Figure 6. Preference Linking using Word Distance

recruiting 151 distinct crowd workers to respond to 12 scenario prompts. Each worker may choose to answer one or more of the prompts, and each answered 1.44 prompts on average. The corpus consists of 217 scenarios, each with a minimum of 150 words, yielding 2,088 sentences and 46,173 words, overall. The annotators, consisting of the first and second authors, independently annotated a sample of 20 scenarios using the coding frame described in Section III-A to yield an inter-rater agreement (Cohen's Kappa) of 0.94. The annotators (first and second authors) annotated the remaining corpus independently to yield a total of 7,661 labeled items.

Table III presents average, relative frequencies per scenario for the labeled items organized by the prompt category (e.g., "Apartment" corresponds to the prompt, "How do you find an apartment?"), the number of scenarios completed by a unique worker in the category, the average number of sentences per scenario in the category, and the average number of labeled items per scenario, separately counted by type: (E)ntity, (M)odifier, (R)elation, and (A)ction, as defined in Section III-A. The last row describes the average frequency *per scenario* for the total number of sentences and labeled items.

Table III
Summary Statistics for Scenario Corpus

Category	Scen #	Sent #	E	M	R	A
Apartment	20	9.9	25.9	4.0	2.4	2.5
Restaurant	18	9.2	18.8	3.2	1.1	2.8
Hiking	20	9.7	27.2	4.2	3.6	3.4
Health Clinic	20	8.8	23.9	3.8	2.5	3.5
Flight Booking	17	8.7	27.8	2.2	1.4	3.0
Social Net	16	9.1	29.8	2.0	1.9	2.3
Movies	16	8.8	24.8	2.1	1.5	3.2
Hotel Booking	17	10.2	32.5	3.2	3.0	2.5
Degree Programs	18	10.4	27.2	3.4	1.9	6.1
Job Hunting	19	10.9	28.2	2.4	1.8	3.9
Travel	18	9.6	27.5	2.2	2.4	3.1
Course Selection	18	9.9	26.2	4.6	1.8	5.7
Total/Avg. Freq.	**217**	**9.6**	**26.6**	**3.1**	**2.1**	**3.5**

In RQ1, we asked "How do stakeholders describe their individual preferences in scenarios they author?" As shown in Table III, the grounded theory introduces four categories of preference words or phrases that stakeholders use to express their preferences: entities (E), modifiers (M), relations (R) and actions (A). Among these categories, *modifier* indicates a category of words that differentiate what stakeholders want of the entity modified. For example, some stakeholders prefer a "good and quiet neighborhood", while others prefer a "lively neighborhood". As a component of the grounded theory, modifiers predict how entity descriptions correspond to alternative

preferences: if an entity has a modifier or can be modified, then a stakeholder can have a preference, and if there are *n* modifiers to an entity, then there could be *n* weights assigned to each modifier to indicate the distribution of stakeholder preferences for that entity. Similarly, the *relation* indicates how stakeholders associate their entities of interest, such as if a "neighborhood" is "close to a grocery store", or "sits in a modern area". Action indicates either a distinguishing aspect of an entity that a stakeholder prefers, or a stakeholder's goal. The discovered theory shows how these categories of preference words coordinate to communicate stakeholder preferences, and further how a requirements analyst can use these categories to discover such preferences, which is a topic that is beyond the scope of this study. As a comprehensive example, in scenario #S0001-R06, the author wrote, "In finding an apartment, first I would make sure that it is a home that is near my job. Then, I must find one in a good and quiet neighborhood. Next, I need to check it out if it is handicap accessible. The place has to be disable friendly where a wheelchair can roam around freely." Herein, we observe that "apartment", "neighborhood", and "wheelchair" are examples of entities to which the author has specific preferences. In addition, "good", "quiet", "handicap accessible", "disable friendly" and "freely" are examples of modifiers, the word "near" indicates a preferential relation, and finally "roam" indicates an preferential action. We found that these four categories are exhaustive, given the Cohen's Kappa score of 0.94, and that no new preference categories were observed.

In this evaluation, we report results using the proportions of true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN) as follows: *precision* is equal to TP / (TP + FP); *recall* is equal to TP / (TP + FN); and F1 Score is the standard harmonic mean of precision and recall. True positives (TP) are tokens that are labeled not as outside (O) in the ground truth that the model classified correctly. False positives (FP) are tokens that are labeled as (O) in the ground truth that the model classified incorrectly as some labels other than (O). True negatives (TN) are tokens that are labeled as (O) in the ground truth that the model classified correctly as (O). False negatives (FN) are tokens that are labeled not as outside (O) in the ground truth that the model classified incorrectly as (O). We now present the results of the app feature survey and automated extraction and linking of preferences from scenarios.

A. Preference Gaps in Existing Systems

As described in Section III-C, the labeled preferences were coded to discover feature categories. In RQ2, we ask "to what extent are the elicited stakeholder preferences satisfied by existing directory services?" To answer this question, we used the discovered features from the coded preferences to survey popular directory service websites in the U.S., and we scored each site for the presence or absence of a specific feature. For website selection, we identified the top twenty results of a Google search using keywords from four of our scenario questions for apartments, restaurants, hiking, and health clinics. We ran the app feature survey on these selected four categories because they are among the most frequently used services, and have a lot of website applications specifically targeting each of them. The summary number of features identified appears in Table IV, which shows the scenario category, the number of labeled preferences (Lab.), the number of unique features (Uniq.), and commonly missing feature categories - feature categories missing from over half of the directory service websites that we surveyed.

Table IV
REQUIREMENTS COVERAGE SURVEY RESULTS

Category	Lab.	Uniq.	Missing Requirements
Apartments	254	11	Commute, Kid-Friendly, Noise, Safety
Restaurant	207	12	Availability, Cleanliness, Coordination, Service
Hiking Trail	410	11	Crowdedness, Difficulty Level, Equipment, Exercise, Weather
Health Clinic	279	10	Schedule Availability, Walk-Ins, Travel Distance, Communications, Care Quality, Affordability

Table V presents the survey results for four popular Apartment finding websites ApartmentFinder.com (F), Apartments.com (A), Rent.com (R) and Zumper.com (Z). Each website provided filters for monthly rent, amenities, wheelchair accessibility (listed as an amenity), and pet-friendliness (distinguishing cats and dogs). Missing requirements identified in scenarios include: commute time to work or the grocery store, kid friendliness, noise from neighbors, and neighborhood safety. While all sites provided mapping features, users would need to individually plot distance to or from work, and conduct additional research to identify area grocery stores, playgrounds, schools, etc.

Table V
REQUIREMENTS COVERAGE FOR APARTMENTS

Feature Category	F	A	R	Z
Accessibility - wheel chair accessible	X	X	X	X
Affordability - monthly rental price	X	X	X	X
Amenities - lists pool, gym, laundry	X	X	X	X
Atmosphere - quiet neighbors	-	-	-	-
Commute - to work, to grocery	-	-	-	-
Kid-friendly	-	-	-	-
Layout - number of bedrooms	X	X	X	X
Layout - upstairs unit	-	-	-	-
Maintenance - apartment unit upkeep	-	-	-	-
Pet-Friendly - allows pets	X	X	X	X
Safety - in safe neighborhood	-	-	-	-

Table VI shows the survey results of the four popular restaurant finding websites OpenTable (O), TripAdvisor (T), Yelp (Y) and Zagat (Z). All sites provided filters for price, cuisine or menu options, location and options for dining in, take-out or delivery. While location was shown, no site provided information on commute time. Three of four sites included reviews and listed kid-friendly and diet options: three

indicated vegetarian, one indicated gluten-free. OpenTable and Yelp provided atmosphere filters related to noise and intimacy, and OpenTable indicated wait time. All sites were missing information about cleanliness and service, and none offered the ability of friends and family to coordinate restaurant selection in real-time.

Table VI
REQUIREMENTS COVERAGE FOR RESTAURANTS

Feature Category	**O**	**T**	**Y**	**Z**
Affordability - total cost of meals	X	X	X	X
Atmosphere - seating, crowded	X	-	X	-
Availability - wait time before seating	X	-	-	-
Cleanliness - of restaurant, bathroom	-	-	-	
Cuisine - menu, regionality and authenticity	X	X	X	X
Diet - food restrictions	X	X	X	-
Diet - friend's diet	-	-	-	-
Kid-friendly - seating, diet for children	X	X	X	-
Location - travel time to restaurant	X	X	X	X
Options - Dine-in, Delivery or Take-out	X	X	X	X
Reviews and ratings	X	X	X	-
Service - personality and timely	-	-	-	-

Table VII shows the survey results for four popular hiking trail finding sites AllTrails.com (A), Cairn (C), Gaia GPS (G), and TrailLink.com (T). All sites provided measures of hiking time or distance, with AllTrails.com satisfying the most features, including tags to indicate scenery, trail amenities, difficulty level, and dog-friendliness. No site included information on trail crowdedness or supplies needed, and only half had weather-related information. Notably, authors expressed many preferences over supplies that could be informed by trail length and weather-related information.

Table VII
REQUIREMENTS COVERAGE FOR HIKING

Feature Category	**A**	**C**	**G**	**T**
Supplies - water, food, rope, etc.	-	-	-	-
Difficulty Level - easy, difficult	X	-	X	-
Weather - sunny, rainy	X	-	X	-
Hiking time - from start to finish	-	X	X	-
Hiking distance	X	X	X	X
Scenery - trail features and amenities	X	-	-	X
Crowdedness - popularity, number of visitors	-	-	-	-
Exercise - suitability for cardio	-	-	X	-
Trail hours - opening and closing times	-	-	-	-
Dog-friendly	X	-	-	-
Reviews - written hiker recommendation	X	-	-	X

Lastly, Table VIII presents the survey results for popular health clinic directory services, including BCBS.com (B), which is a prominent U.S. health insurer, HealthGrades.com (H), ShareCare.com (S), and WebMD.com(W). Features to check schedule availability, including walk-in availability, were missing, as well as travel distance, care quality, and affordability.

In answer to RQ2, "to what extent are the elicited stakeholder preferences satisfied by existing directory services," Tables IV-VIII illustrate that while some app websites include features that satisfy many stakeholder preferences (e.g., OpenTable, AllTrails.com, HealthGrades), most websites are not comprehensives and illustrate gaps or exhibit missing features. A notable pattern across stakeholder preferences is a heavy reliance on user-provided reviews and ratings. This indicates that many directory services have "plateaued" as limited features force users to rely on unstructured information to make selections. In contrast, the three services (OpenTable.com, AllTrails.com, and HealthGrades.com) clearly satisfy more preferences than their peer sites by addressing more personalized needs.

Table VIII
REQUIREMENTS COVERAGE FOR HEALTH CLINIC

Feature Category	**B**	**H**	**S**	**W**
Availability - scheduling, walk-ins	X	X	X	X
Insurance - acceptance, coverage	X	X	X	X
Distance - travel time, distance in miles	X	-	-	-
Specialities - appropriate for type of issue	X	X	X	X
Communications - bedside manner, respect	-	X	-	X
Care quality	-	-	-	-
Wait time - walk-ins, scheduling backup	-	X	-	X
Affordability	-	-	-	-
Doctor gender	X	X	X	-
Reviews	-	X	X	X

B. Automated Preference Extraction

We now present results of three classifiers to automatically extract stakeholder preferences from scenarios: (1) conditional random fields trained on word distributions; (2) random forest trained on typed dependencies; and (3) transformer trained using BERT word embeddings. In RQ3, we asked "what natural language features are best suited to automatically extract preferences from scenarios?" Each of the three selected models use different natural language features to predict preferences. In CRF, the author's sentences are encoded as word sequences and label sequences, which form a chain comprised of the last label and the next label. In this respect, CRF is can be used to find multi-label phrases by conjoining labels. Typed dependencies use binary relations among word based on their grammatical functions (e.g., determiners, adjectives, direct objects, etc.) Where CRF is grammar-agnostic, using type dependencies tests whether grammatical information is predictive of which word sequences indicate stakeholder preferences. Finally, BERT uses word embeddings that encode information between words and sentences. Each word corresponds to a vector that encodes statistical relations among words and sentences that can be used to predict semantic relationships. By comparing each of these models, we are interested in which of these feature encodings are best suited to extract preferences. We evaluated the three classifiers in two ways: (1) A *seen evaluation* that trains the model on 4/5 of all the scenarios randomly selected from each category, and tests on the remaining 1/5 of the scenarios from each category. The process is repeated 10 times, each time randomizing

the scenarios used for training and test, and the results are averaged. (2) An *unseen evaluation* that trains the model on 11/12 scenario categories, and tests on the remaining 1/12 scenario categories not seen during training. This process is repeated 12 times using different combinations of training and test data, and the results are averaged. For the transformer, we sub-divide the training data in both seen and unseen evaluations as follows: we use 80% of the training data for training, and 20% for validation, which includes tuning hyper-parameters and choosing the number of epochs for training. Table IX presents the precision, recall and F1-score for each approach. The highest scores are in bold.

We observe that the BERT-Transformer model gives the best weighted overall average F1 score, as well as performing the best in classifying modifiers, relations and actions among all experiments with one exception: the Random Forest classifier performs slightly better on entity classification in seen domains.

C. Automated Preference Linking

The preference linker, described in Section III-E, was evaluated by both authors by developing a linking corpus that consists of one sentence for each labeled non-entity from the ground truth preference corpus, in addition to the one entity to which the non-entity is meaningfully attached. Those actions that describe goals linked to the author instead of to an entity were excluded from the ground truth. We did the evaluation on the ground-truth dataset instead of on our models, in order to avoid the influence of model errors on our preference linker. Next, the linking technique was applied to the itemized sentences in the preference corpus that contains all the ground truth annotations, and word distance was computed for each entity in the sentence, with regard to each non-entity in the same sentence. The entity with the shortest distance, which became the prediction result, was compared to the linking corpus: if the entity matched, the link was correct, if not, the link was incorrect. This evaluation results in the best-case performance, which would be reduced by any errors produced by the best performing classifier. The linking technique was evaluated using accuracy, which is the ratio of correct links to the total number of non-entities to be linked.

Table X presents the average number of linked preferences per scenario (Avg Prefs), the total number of non-entity labels correctly linked to entity labels (Correct), the total number of non-entity labels (Total), and the Accuracy for each non-entity label type, in addition, to the overall summative performance. Each authored scenario includes on average 6.2 preferences across each of the non-entity link types.

Table XI presents examples of correctly linked preferences to illustrate the results that developers could use as outputs of the overall approach.

V. Discussion

We now discuss the error analysis, privacy-sensitive preferences, and the approach's practical application.

A. Extraction Error Analysis

We examined sources of extraction error by developing Model A, which is trained on eleven of the question categories and tested on the remaining category, Apartments. Table XII presents the precision, recall, true positives (TP), false positives (FP), and false negatives (FN) for Model A for each of the three classifiers. To identify the error sources for all models, we randomly sampled 20 sentences from the test dataset and compared predicted and ground-truth labels for all sentence tokens. Detailed analyses are discussed below.

1) CRF Model Error: The Conditional Random Fields (CRF) model applied to the 20 sentences yielded 608 tokens, among which 56 (9.2%) are errors. Among the errors, 32/56 (57.1%), 2/56 (3.6%), 4/56 (7.1%), 1/56 (1.8%) are entities, modifiers, relations and actions, respectively, wrongly classified by the CRF model. The remaining 17/56 (30.4%) errors are labeled as "outside" in ground-truth but not predicted so, which are false positives (FPs). We observe 23/56 (41.1%) are span-related errors where a label covered a portion of an entity. For instance, for the phrase "available units", where each of the 2 tokens "available" and "units" should be labeled as an entity, the model only labeled the token "units" as an "entity". 6/56 (10.7%) of the errors are English conjunction errors, in which the model labels one or more elements in the list, but misses other list elements. For instance, in the phrase "right school and degree program", the model correctly labels "right school" and misses "degree program". 2/56 (3.6%) of the errors are goal-action errors, where actions indicating a goal of the author are missed by the model. For instance, the word "live" in the phrase "I want to live by myself" should be labeled as an action but is missed by the classifier. Finally, 3/56 (5.4%) of the errors are mistakes made in the ground-truth, where the independent investigators should have labeled items that were missed.

2) Random Forest Model Error: The Random Forest (RF) model applied to 20 randomly selected sentences yielded 2,436 tokens, among which 93 were errors, to yield an error rate of 3.8%. Among these errors, 7/93 (7.5%), 39/93 (41.9%), 5/93 (5.4%) and 11/93 (11.8%) are entities, modifiers, relations and actions, respectively, wrongly classified by the RF Model. The remaining 31/93 (33.3%) errors are false positives (FPs).

3) BERT-Transformer Model Error: The BERT-Transformer model applied to the 20 sentences yielded 595 tokens, among which 25 tokens are errors, to yield an error rate of 4.1%. Among these errors, 3/25 (12.0%), 0/25 (0%), 1/25 (4.0%), 1/25 (4.0%) are entities, modifiers, relations and actions, respectively, wrongly classified by the BERT-Transformer model. The remaining 20/25 (80.0%) errors are labeled as "outside" in ground-truth but not so in the predictions, which are false positives (FPs). Following the explanations as in Section V-A1, we observe that 7/25 (28.0%) of the errors are span-related errors; 1/25 (4.0%) are goal-action errors; and 2/25 (8.0%) are ground-truth mistakes made in the ground-truth.

4) Error Correction: Several classifier errors may be fixed with post-processing to improve performance. For example,

Table IX
SUMMARY STATISTICS FOR THREE CLASSIFIERS WITH SEEN AND UNSEEN DOMAINS

Evaluation on Scenarios from Previously Seen Domains									
	Conditional Random Fields			Random Forest			BERT-Transformer		
Label	**Prec.**	**Recall**	**F1**	**Prec.**	**Recall**	**F1**	**Prec.**	**Recall**	**F1**
entity	0.8146	0.7703	0.7918	0.8581	**0.9202**	**0.8880**	**0.8645**	0.9017	0.8826
modifier	**0.7715**	0.5247	0.6239	0.7489	0.6097	0.6720	0.7074	**0.7544**	**0.7294**
relation	**0.7496**	0.4641	0.5710	0.2818	0.0953	0.1405	0.6914	**0.7128**	**0.7008**
action	**0.6977**	0.4077	0.5117	0.6616	0.2796	0.3929	0.6362	**0.6655**	**0.6491**
Weighted, overall average	0.8008	0.6920	0.7424	0.7796	0.7730	0.7654	**0.8160**	**0.8525**	**0.8338**
Evaluation on Scenarios from Previously Unseen Domains									
	Conditional Random Fields			Random Forest			BERT-Transformer		
Label	**Prec.**	**Recall**	**F1**	**Prec.**	**Recall**	**F1**	**Prec.**	**Recall**	**F1**
entity	0.7906	0.6702	0.7240	0.8301	**0.9021**	0.8638	**0.8661**	0.8925	**0.8789**
modifier	**0.7312**	0.5182	0.6037	0.5786	0.4662	0.5099	0.7101	**0.7599**	**0.7280**
relation	0.6197	0.3826	0.4668	0.2321	0.0950	0.1322	**0.6579**	**0.6808**	**0.6640**
action	**0.6266**	0.3817	0.4668	0.2509	0.0688	0.1056	0.5908	**0.6423**	**0.6059**
Weighted, overall average	0.7683	0.6117	0.6798	0.7150	0.7331	0.7139	**0.8106**	**0.8435**	**0.8262**

Table X
ACCURACY OF STATIC PREFERENCE LINKER

Non-Entity	Avg Prefs	Correct	Total	Accuracy
Modifier	3.26	481	557	0.8636
Relation	2.27	369	382	0.9660
Action	2.51	342	384	0.8906
Overall	6.21	1192	1323	0.9010

Table XI
EXAMPLES OF LINKED PREFERENCES

Category	Example Phrases
Apartment	[**apartment**] that is [**close**] to my [workplace]
Hiking	long and [**difficult**] of a [**hike**]
Hiking	swimming [**area**] [**along**] the [hike]
Hiking	a nice [**view**] where I could [**relax**]
Health Clinic	[**clinic**] that is [**professional**]
Flight Booking	boarding [**pass**] that [**allows**] you to get on the [flight first]
Movies	[**movie**] doesn't [**make**] me laugh
Movies	watch a [**movie**] [**in**] the [theater]
Degree Programs	how [**respected**] the [**programs**] are

conjunction errors may be detectable by finding *conj* dependencies linked to the labeled entities. Similarly, span errors may be remediated by checking neighboring parts of speech (PoS) against inclusion and exclusion criteria (e.g., entities described by noun phrases may contain adjectives, nouns and conjunctions, but not other PoS). Furthermore, we acknowledge that scenarios we use to train our models are written by human authors and thus may contain errors in grammar, spelling, punctuation, etc. Although we have taken precautions by rejecting scenarios that contained more than five spelling, capitalization or grammar errors, pre-processing steps may still be taken, manually or automatically, to correct these errors, which may in turn boost the model performance. We leave the pre-processing and post-processing, along with other possible techniques to further improve the performance of the classifiers, to future work.

5) Model Comparison: We observe from Table IX that the BERT-Transformer model gives the best weighted overall average F1 score, as well as performing the best in classifying modifiers, relations and actions among all experiments with one exception: the Random Forest classifier performs slightly better on entity classification in seen domains. This particular exception itself is hard to empirically explain from the empirical result alone. While one can acquire the weights of tokenized words used by RF to predict the class labels, these weighted tokens are also rarely good explanations for predictions that involve complex lexical and syntactic structures in text. Since this performance difference is small, it may not indicate a reliable superiority of the Random Forest classifier on entity classification in seen domains. Furthermore, we observe the trend that the CRFs model tends to have a decent precision yet a low recall. We conjecture that this may be because the CRFs model is a "picky" model that makes careful predictions - it is capable of tagging the predicted positive results with correct labels, yet suffers in distinguishing between positive and negative results. We also observe that overall, the models' performance on entity is better than that on modifier, relation, and action. This may be caused by each label's unequal representation in the dataset - entities generally have more training samples than the other labels. Moreover, when trained on a Precision 3640 Tower Machine's CPUs, the CRFs model and the Random Forest model are both quicker to train than the BERT-based model, but all three are able to finish the training within a reasonable amount of time. Specifically, the BERT-based model takes around 22 minutes to train for 15 epochs, while the CRFs model and the Random Forest model both finish training within 5 minutes.

B. Linking Labels to Preferences

The preference linker yielded 131 errors out of 1323 expected preference phrases (9.9%), as presented in Table X. An analysis of these 131 errors identified three primary causes:

- *leftward bias* (32%) is a bias to find expected entities left of the non-entity label, e.g., "[apps] provide [fast], reliable and free calling [facilities]" where "fast" should link to "facilities," but the linker chooses "apps."
- *intermediating subject* (6%) is where the labeled entity occurs left of the incorrect entity, which is the subject of the action, e.g., "[place] that [friends] really [enjoyed]" in which "place" is the correct entity, and "friends" is the intermediating subject.
- *intermediating clause* (59%) is where the labeled entity occurs left of one or more clauses in a list, which include incorrect entities, e.g., "[providers] are in [network] and

Table XII
SUMMARY STATISTICS FOR THREE CLASSIFIERS USING MODEL A VARIANT

	Conditional Random Fields					Random Forest					BERT-Transformer				
Label	**Prec.**	**Recall**	**TP**	**FP**	**FN**	**Prec.**	**Recall**	**TP**	**FP**	**FN**	**Prec.**	**Recall**	**TP**	**FP**	**FN**
entity	0.7868	0.7110	369	100	150	0.7831	0.9268	1823	476	144	0.8710	0.9364	740	68	27
modifier	0.7746	0.6790	55	16	26	0.6522	0.5305	165	86	146	0.7245	0.8765	81	19	11
relation	0.6389	0.4792	23	13	25	0.0833	0.0196	1	9	50	0.5738	0.7292	36	25	13
action	0.5200	0.5200	26	24	24	0.1356	0.0586	16	94	257	0.5063	0.800	41	38	12

[within] 10 miles of my home" in which the relation word "within" links to "providers," yet "network" is in the intermediating clause "in network."

Among all error types observed, 59% of the error is due to intermediating clauses, 32% is due to leftward bias, and 6% to intermediating subjects. A more robust technique, such as typed dependencies or constituencies, which link words across clauses, may improve these results in future work.

Moreover, the preference linker is one simple method to re-establish the context for the results from preference coding, but we acknowledge that it is not complete. More context is likely missing. We leave other possible context restoration techniques to future work.

C. Privacy-sensitive Preferences

Because preferences are personal to individual users, they can concern privacy, such as income, health status, marital status, and presence/absence of children, and app features that support their collection should also protect this data as personal information. Preferences about affordability include low-income, such as "avoid living check to check." Mid- and high-income preferences covered apartment amenities, such as "in-unit laundry" or "gym," and airline flyers willing to spend money to avoid "poor meals" or "sit closer to the front of the plane." In the restaurant category, authors expressed preferences for dietary restrictions due to "digestive disorder[s]," and in the hiker trail category, authors described preferences for "easier, shorter hike[s]" versus "long" and "strenuous" hikes "in steep conditions." In restaurants, hiking, movies, and travel, authors referred to their marital status using words such as "girlfriend," "husband" and "wife," when stating their preferences and how they make decisions. Finally, authors further referred to entities that are "kid friendly," and "good for children," and "expert in children medics."

Privacy-sensitive preferences introduce additional concerns for elicitation: (1) privacy-sensitive preferences are generally exclusionary; and (2) stakeholders may be less willing to disclose these preferences. *Exclusionary preferences* are those that stakeholders hold at the exclusion of holding other preferences. Because a stakeholder is low-income, their preferences for affordability would be at the exclusion of preferences for luxury. Preferences for kid-friendly services will exclude services where children cannot be easily accommodated. Exclusionary preferences based on stakeholder traits require special attention during author sampling to ensure broad representation across traits. This may improve scenario completeness by asking follow-on questions to elicit exclusionary preferences. Because stakeholders may be less willing to disclose privacy-sensitive preferences, techniques such as *seeding the writing prompt* or *eliciting targeted quality descriptions* can be used, when it is known that a stakeholder is in a privacy-sensitive class. To recognize if a stakeholder is in such a class, one can conduct a pretest before scenario authoring.

D. Applicability to Requirements Practice

This work is significant because it enables developers to extract preferences from scenarios, which can then be used to identify new or missing requirements as illustrated in the app survey. Directory service developers can now obtain stakeholder preferences as follows: 1) use the existing scenario dataset, or if the directory service category is not covered in that dataset, collect new scenarios from stakeholders. Stakeholders can be existing users, or workers hired from Amazon Mechanical Turk. Special attention may be needed to assess worker experience with the domain, before eliciting scenarios. The number of scenarios needed depends on the diversity of individual preferences. For four domains, our results show that 16-20 scenarios yielded 15-27 feature categories, among which 49.7% were missing in popular sites. 2) Apply our model and tools to automatically extract preference labels from the scenarios. 3) Apply the preference linker to produce preference phrases from preference labels.

VI. THREATS TO VALIDITY

Construct validity refers to whether we are measuring what we believe we are measuring [72]. The construction of the ground truth annotations requires a carefully designed coding frame that is applicable by any investigator and able to achieve similar results across applications. The preference types and the scenarios annotations must be consistent and repeatable. Moreover, the feature categories in the App Feature Survey are the authors' interpretations or categories of the user-stated preferences, which have not been cross-validated with those scenario authors. To address the coding threat, the authors employed a three-step annotation process where they compared and converged their annotations, while computing the inter-rater reliability to yield a high, above-chance agreement to reveal specific sources of disagreement. The same validated ground truth corpus was used to separately evaluate the results of the app survey, classifier study, and linking study. Similarly, the app features that were identified from the ground truth corpus were reviewed by both authors, independently, prior to assessing the directory services for gaps.

Internal validity refers to validity of analyses and conclusions drawn from the data [72]. To reduce this threat, the authors carefully reviewed and updated the heuristics used to assign labels to the scenarios. In the app survey, the investigators reviewed the directory service websites for functions that

match features identified from stakeholder preferences. This procedure depends on user familiarity with the services, and thus app features could be present but unaccounted for. To address this threat, the first author validated the second author's categorization and independently examined the features, and both authors agreed on the survey results. To ensure results comparability among classifiers, the authors used the same dataset partitions to evaluate the seen and unseen domains, and the same performance measures.

External validity refers to the generalizability of results [72]. Preferences are unique to individual stakeholders, and preferences described by scenario authors will vary based on author personality, interests and writing styles. As a grounded theory, the preference types are only valid for this dataset, and generalizability to other domains is unknown. We chose the directory services domain because Internet users spend large amounts of time searching for information online, thus improvements to these services could save users valuable time, and because personalizing these services requires higher quality data on what factors affect user preferences, thus directory services is an ideal fit to study preferences. In addition, personalization as a software feature is cross-cutting and appears in other domains, e.g., shopping includes features to search for products, and entertainment includes features to search for movies and music. Thus, while this domain is seemingly narrow, we believe better requirements models of user preferences will be beneficial to understanding other domains. To support generalizability, we collected scenarios from 12 different directory service domains and from 157 different authors. These authors were drawn from the AMT worker population, which is 57% female, skewed younger, with a relatively similar racial and economic distribution to the U.S. population [42]. We limited workers to US location, which correlates with English proficiency. To our knowledge, the underrepresented demographics do not use English differently to express preferences, but they may express different preferences. Statistical models risk overfitting the training data, where performance drops considerably as models are applied to unseen data. To address this threat, we studied multiple domains in two configurations: seen domains, wherein the model predicts labels for new scenarios from domains seen in training data; and unseen domains, where models predict labels for new scenarios from domains excluded from training data. While the preference linker does not rely on machine learning, the reported accuracy is dependent on sentence type: longer, more grammatically complex sentences may affect linker performance. To understand sources of error and their affect on generalizability, we analyzed errors in a random sample of 20 sentences for each classifier and the linker.

VII. Conclusion

In this paper, we conducted studies to elicit stakeholder preferences using scenarios wherein users describe their goals for using directory services to find entities of interest, such as apartments, hiking trails, etc. With greater understanding of preferences, developers can enhance software with features that better satisfy such preferences. Our results indicate that feature support for preferences varies considerably in popular directory services, and that 49.7% of preferences that were identified were not satisfied by these sites, requiring users to resort to using other tools to support their decision-making.

In addition, we studied ways to automatically extract preferences from scenarios using named entity recognition based on three separate approaches. The BERT-based transformer performed best with an average overall 81.1% precision, 84.4% recall and 82.6% F1-score evaluated on scenarios from unseen domains. Finally, we describe a static preference linker that links extracted entities into preference phrases with 90.1% accuracy. Based on this pipeline, we believe developers could use our BERT-based model and preference link to identify stakeholder preferences from scenarios. The preferences could then be used to identify gaps and to improve how services satisfy stakeholder preferences by addressing those gaps with new or improved app features. The dataset and supporting tools are available online [73].

Acknowledgment

This paper was supported in part by NSF Awards #2007298 and #1453139.

References

[1] R. Artstein, M. Poesio. "Inter-coder agreement for computational linguistics," *Computational Linguistics*, 34(4): 555-596, 2008.

[2] S. Abualhaija, C. Arora, M. Sabetzadeh, L. C. Briand, M. Traynor. "Automated demarcation of requirements in textualspecifications: a machine learning-based approach," *Empirical Software Engineering*, 25:5454–5497, 2020.

[3] C. Arora, M. Sabetzadeh, L. Briand, F. Zimmer, "Extracting domain models from natural-language requirements: approach and industrial evaluation", In Proc. *ACM/IEEE* 19^{th} *International Conference on Model Driven Engineering Languages and Systems (MODELS '16)*, 2016.

[4] C. Arora, M. Sabetzadeh, L.C. Briand, "An empirical study on the potential usefulness of domain models for completeness checking of requirements." *Empirical Software Engineering*, 24: 2509–2539, 2019.

[5] B. Boehm, R. Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley, 2003.

[6] W. Bousfield, W. Barclay, "The relationship between order and frequency of occurrence of restricted associative responses," *Journal of Experimental Psychology*, 40: 643-647, 1950.

[7] D.D. Brewer. Supplementary interviewing techniques to maximize output in free listing tasks. *Field methods*, 14(1): 108-118, 2002.

[8] M. Broy, "Domain Modeling and Domain Engineering: Key Tasks in Requirements Engineering." *Perspectives on the Future of Software Engineering*, 2013.

[9] S. Brin, "Extracting Patterns and Relations from the World Wide Web", In *Proc. Conference of Extending Database Technology. Workshop on the Web and Databases*, 1998.

[10] K. Charmaz, Constructing Grounded Theory, 2nd ed. SAGE, 2014.

[11] J. Chiu, E. Nichols, "Named Entity Recognition with Bidirectional LSTM-CNNs", *Transactions of the Association for Computational Linguistics*, 4:357-370, 2016.

[12] C. Cortes, V. Vapnik, "Support-vector networks". *Machine learning, 20(3), 273-297*, 1995.

[13] J. Devlin, M. Chang, K. Lee, K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv:1810.04805*, 2018.

[14] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir and S. Çevikol, "Requirements Classification with Interpretable Machine Learning and Dependency Parsing," In Proc. *IEEE* 27^{th} *International Requirements Engineering Conference (RE)*, pp. 142-152, 2019.

[15] A. Dardenne, A. van Lamsweerde, S. Fickas. "Goal-directed requirements acquisition," Science of computer programming, 20(1-2): 3-50, 1993.

[16] S. M. Davis. *Future Perfect*. Adison-Wesley, Reading, MA, 1987.

[17] C. Detweiler, M. Harbers, "Value Stories: Putting Human Values into Requirements Engineering", *REFSQ Workshops*, 2-11, 2014.

[18] J. Finkel, T. Grenager, C. Manning, "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling", *roceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pp. 363-370, 2005.

[19] J. Fink, A. Kobsa. "A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web." *User Modeling and User-Adapted Interaction* 10: 209-249, 2000.

[20] A.J. Franco, "Requirements elicitation approaches: A systematic review," *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS), pp. 520-521, doi: 10.1109/RCIS.2015.7128917*, 2015.

[21] M. Garima, M. Cevik, Y. Khedr, D. Parikh, and A. Basar, "Named Entity Recognition on Software Requirements Specification Documents." *Canadian Conference on AI*, 2021.

[22] J. Grudin, J. Pruitt, "Personas, Participatory Design and Product Development: An Infrastructure for Engagement", *Proceedings of Participation and Design Conference (PDC2002), page 144-161*, 2002.

[23] M. Glinz, "On Non-Functional Requirements," In Proc. *15th International Conference on Requirements Engineering*, pp. 21-26, 2007.

[24] B. Gonzalez-Baixauli, J. C. S. do Prado Leite, M. A. Laguna, "Eliciting Non-Functional Requirements Interactions Using the Personal Construct Theory," *14th IEEE International Requirements Engineering Conference (RE'06), pp. 347-348, doi: 10.1109/RE.2006.18*, 2006.

[25] J. Horkoff, F.B. Aydemir, E. Cardoso, et al. "Goal-oriented requirements engineering: an extended systematic mapping study", *Requirements Eng 24, 133–160*, 2019.

[26] G. B. Herwanto, G. Quirchmayr and A. M. Tjoa, "A Named Entity Recognition Based Approach for Privacy Requirements Engineering," *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW), pp. 406-411*, 2021.

[27] Z. Huang, W. Xu, K. Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR, abs/1508.01991*, 2015.

[28] D. Jurafksy and J. Martin. Speech and Language Processing, 2nd ed. Prentice Hall, 2008.

[29] M. Jackson, "The World and the Machine." In Proc. *17th International Conference on Software Engineering*, pp. 283–292, 1995.

[30] R. Jiang, R.E. Banchs, H. Li, "Evaluating and Combining Named Entity Recognition Systems", *Proceedings of the Sixth Named Entity Workshop*, 2016.

[31] N. Jha, A. Mahmoud. "Mining non-functional requirements from App store reviews," *Empirical Software Engineering*, 24: 3659–3695, 2019.

[32] N. Janpitak, C. Sathitwiriyawong and P. Pipatthanaudomdee, "Information Security Requirement Extraction from Regulatory Documents using GATE/ANNIC," *7th International Electrical Engineering Congress (iEECON), Hua Hin, Thailand, 2019, pp. 1-4*, 2019.

[33] Z. Kurtanović and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning", In Proc. *IEEE International Requirements Engineering Conference (RE),* pp. 490-495, 2017.

[34] R. Kohavi, R. Longbotham, D. Sommerfield, R.M. Henne. "Controlled experiments on the web:survey and practical guide." *Data Mining & Knowledge Discovery*, 18:140–181, 2009.

[35] A. Kumar. From mass customization to mass personalization: a strategic transformation." International *Journal Flexible Manufacturing Systems*, 19:533-547, 2007.

[36] S. Lim, A. Henriksson, J. Zdravkovic, "Data-Driven Requirements Elicitation: A Systematic Literature Review", *SN Computer Science. 2, 16*, (2021)

[37] J.R. Landis, G.G. Koch. "The measurement of observer agreement for categorical data." *Biometrics*, 33(1):159—174, 1977.

[38] C. Li, L. Huang, J. Ge, B. Luo, V. Ng, "Automatically classifying user requests in crowdsourcing requirements engineering." *J. Syst. Softw.*, 138: 108-123, 2018.

[39] S. Liaskos, S. A. McIlraith, S. Sohrabi, J. Mylopoulos, "Integrating Preferences into Goal Models for Requirements Engineering," *2010 18th IEEE International Requirements Engineering Conference*, pp. 135-144, 2010.

[40] S. Liaskos, S. A. McIlraith, S. Sohrabi, J. Mylopoulos, "Representing and reasoning about preferences in requirements engineering", *Requir. Eng. 16*, 3: 227-249, 2011.

[41] J. Lafferty, A. McCallum, F.C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data, 2001.

[42] L. Litman, J. Robinson, "Conducting Online Research on Amazon Mechanical Turk and Beyond," *SAGE*, 2020.

[43] M.-C. de Marneffe, B. MacCartney, C. D. Manning. "Generating Typed Dependency Parses from Phrase Structure Parses," *International Conference on Language Resources and Evaluation*, 2006.

[44] J. Mitchell, M. Lapata. "Vector-based models of semantic composition." In Proc. Association for Computational Linguistics, pp. 236–244, 2008.

[45] D. Mougouei, H. Perera, W. Hussain, R. Shams, J. Whittle. "Operationalizing Human Values in Software: A Research Roadmap," *ACM Fnds. Soft. Engr.*, pp. 780-784, 2018.

[46] P.K. Murukannaiah, N.Ajmeri, M.P. Singh. "Acquiring Creative Requirements from the Crowd." *IEEE* 24^{th} *Int'l Reqts. Engr. Conf.*, pp. 176-185, 2016.

[47] W. Maalej, H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," *2015 IEEE 23rd International Requirements Engineering Conference (RE), pp. 116-125, doi: 10.1109/RE.2015.7320414*, 2015.

[48] J. Mylopoulos, L. Chung, B. Nixon, "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach". IEEE Trans. Soft. Eng., 18(6): 483-497, 1992.

[49] N. Niu, S. Easterbrook, "So, You Think You Know Others' Goals? A Repertory Grid Study," *IEEE Software, vol. 24, no. 2, pp. 53-61, doi: 10.1109/MS.2007.52*, 2007.

[50] D. Nadeau, P. Turney, S. Matwin, "Unsupervised Named Entity Recognition: Generating Gazetteers and Resolving Ambiguity". In *Proc. Canadian Conference on Artificial Intelligence*, 2006.

[51] OpenTable, Inc. "Global Fast Facts," Q4 2019. https://press.opentable.com/static-files/3f990a9c-0702-4496-9b80-5b90198d056a

[52] C. Pacheco, I. Garcia, M. Reyes. "Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques", *IET Software*, 2018.

[53] N. Patil, A. Patil, B.V. Pawar, "Named Entity Recognition using Conditional Random Fields", *Procedia Computer Science, Volume 167*, 2020,

[54] D. Peppers, M. Rogers. The One to One Future: Building Relationships One Customer at a Time. Currency Doubleday. 1993.

[55] M. Praznik, "AllTrails Celebrates 1 Million Paid Subscribers" PR Newswire, 26 Jan 2021.

[56] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, C.D. Manning. "Stanza: A Python Natural Language Processing Toolkit for Many Human Languages." *Association for Computational Linguistics (ACL) System Demonstrations*, 2020.

[57] T. Quirchmayr, B. Paech, R. Kohl, H. Karey and G. Kasdepke, "Semi-automatic rule-based domain terminology and software feature-relevant information extraction from natural language user manuals", *Empirical Software Engineering*, 23: 3630–3683, 2018.

[58] M. Robeer, G. Lucassen, J. M. E. M. van der Werf, F. Dalpiaz and S. Brinkkemper, "Automated Extraction of Conceptual Models from User Stories via NLP," In Proc. *IEEE* 24^{th} *International Requirements Engineering Conference (RE)*, pp. 196-205, 2016.

[59] J. Saldaña. The Coding Manual for Qualitative Researchers, *SAGE Publications*, 2012.

[60] Y. Shen, T. Breaux, "Domain Model Extraction from User-authored Scenarios and Word Embeddings," *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW), pp. 143-151, doi: 10.1109/REW56159.2022.00036*, 2022.

[61] J. Schafer, J. Konstan, J. Riedl. "Recommender systems in electronic commerce." In Proc. *ACM Conference on Electronic Commerce (EC-99)*. ACM. pp. 158-166., 1999.

[62] A. Sleimi, N. Sannier, M. Sabetzadeh, L. Briand and J. Dann, "Automated Extraction of Semantic Legal Metadata using Natural Language Processing," In Proc. *IEEE* 26^{th} *International Requirements Engineering Conference (RE)*, pp. 124-135, 2018.

[63] M. Soeken, C. B. Harris, N. Abdessaied, I. G. Harris and R. Drechsler, "Automating the translation of assertions using natural language processing techniques," In Proc. *Forum on Spec. & Design Lang. (FDL)*, pp. 1-8, 2014.

[64] L. Teixeira, C. Ferreira, B. S. Santos, "User-centered requirements engineering in health information systems: A study in the hemophilia field", *Computer methods and programs in biomedicine*, 106(3):160-174, 2012.
[65] Y. Tsuruoka, J. Tsujii, "Boosting Precision and Recall of Dictionary-Based Protein Name Recognition", In *Proc. Conference of Association for Computational Linguistics. Natural Language Processing in Biomedicine*, 13:41-48, 2003.
[66] M. Tavakoli, L. Zhao, A. Heydari, G. Nenadić, "Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools", *Expert Systems with Applications, Volume 113, 2018, Pages 186-199, ISSN 0957-4174*, 2018
[67] I. Yamada, A. Asai, H. Shindo, H. Takeda, Y. Matsumoto, "LUKE: Deep Contextualized Entity Representations withEntity-aware Self-attention", *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
[68] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, I. Polosukhin. 2017. "Attention is all you need", *Advances in Neural Information Processing Systems, pages 6000–6010*, 2017.
[69] A. van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley, 2009.
[70] S. Victor, L. Debut, J. Chaumond, T. Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." *ArXiv abs/1910.01108*, 2019.
[71] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac et al. "Huggingface's transformers: State-of-the-art natural language processing." *arXiv:1910.03771*, 2019.
[72] R. K. Yin, Case study research: Design and methods, vol. 5. Sage, 2009.
[73] Data and tools for our work: https://figshare.com/articles/software/Stakeholder_Preference_Extraction_from_Scenarios_-_Code_and_Datasets/24431527

Yuchen Shen received the bachelor's degree in Computer Science and Mathematics from Cornell University in 2020. She is currently working toward the PhD degree with Carnegie Mellon University (CMU). Her work focuses on software requirements engineering and natural language processing.

Travis Breaux received his PhD from North Carolina State University in 2009, before joining the School of Computer Science at Carnegie Mellon University (CMU) in 2010, where he is now an Associate Professor. He is the Director of the CMU Requirements Engineering Lab, where he leads research in developing methods and tools for designing trustworthy systems. This includes measuring system conformance to privacy and security policies, standards and laws. His research papers have been nominated for and received best paper awards, including a nomination for the most influential paper in IEEE RE 2016. His work to quantify the relationship between privacy risk and ambiguity expressed in privacy policy was cited as one among several influences in the drafting of the California Consumer Privacy Act (CCPA).