Embeddings-Assisted Requirements Extraction and Elicitation

Yuchen Shen

CMU-S3D-25-108

July 2025

Software and Societal Systems
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Travis Breaux (Chair)
Christian Kästner
Bogdan Vasilescu
Fabiano Dalpiaz (Utrecht University)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Software Engineering

This material is based upon work supported in part by National Science Foundation (NSF) Awards #2007298, #1453139, and #2217572. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: requirements engineering, machine learning, natural language processing, large language models

To my family and friends

Abstract

Companies leverage personalization techniques 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 influence their choices when they use directory services, and these have largely been overlooked or ignored in both the scientific research and the state-of-the-art practices. Specifically, 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 the realm of requirements engineering (RE), requirements analysts endeavor to gather, comprehend, and prioritize requirements, with an important focus on stakeholder preferences and needs, employing diverse requirement elicitation techniques. Advances in Machine Learning (ML) and Natural Language Processing (NLP) have revolutionized the way people understand and interact with natural language, and opened up new opportunities to enhance and automate various facets of requirements engineering, including stakeholder requirements elicitation. These technologies have enabled the analysis of vast amounts of textual data that were previously impractical to process manually, hence improving the accuracy and efficiency of understanding stakeholder needs and preferences. Now, organizations have the potential to automatically extract and prioritize requirements from diverse sources that contain large amounts of natural language, which not only accelerates the requirements engineering process but also reduces the likelihood of human errors, leading to more reliable and user-centered software solutions. Additionally, it allows for continuous feedback loops and real-time updates, ensuring that the project stays aligned with evolving stakeholder demands.

Despite the benefits, there are still a lot left to explore about how the emerging new technologies may be used to aid in and improve both domain knowledge modeling and requirements acquisition. Human access to domain knowledge is challenging in that such knowledge is often tacit and specialized. Requirements elicitation practices often see imbalances in domain knowledge between requirements analysts and stakeholders. This thesis aims address such balances, to explore the potential of embeddings-assisted techniques to enhance stakeholder preference extraction and elicitation practices, particularly by utilizing semantic information encoded in natural language embeddings to supplement gaps in stakeholder knowledge, guide elicitation with the obtained knowledge, and in turn support requirements acquisition. We believe natural language embeddings can be used to improve: 1) domain knowledge modeling: identifying and obtaining requirement-related domain elements; and 2) guided elicitation: given requirement-related domain elements, use them to refine requirements artifacts, such as interviews. Specifically, we demon-

strate the following embeddings-assisted preference extraction and elicitation methods: 1) domain knowledge modeling with user-authored scenarios: we research on how stakeholder preferences are expressed in text scenarios and report our success in extracting and modeling domain knowledge from scenarios using classifiers and linkers; 2) domain knowledge modeling with MLM: we study the efficacy of identifying and obtaining requirement-related domain knowledge from word embeddings using a BERT-based Masked Language Model (MLM), with the aim of discovering missing relationships for requirements, and report our success in discovering associated actors, actions, and modifiers for constructing the domain model; 3) guided elicitation for interviews: we study a method to refine requirements artifacts, in this case interviews, by guiding real-time requirements elicitation interviews with domain knowledge extracted from MLM, and report an improvement on elicitation outcome for both eliciting more concepts and eliciting more specific concepts; 4) interview question generation: we describe a framework outlining common interviewer mistake types and for evaluating follow-up question quality, and conduct studies to test the capability of GPT-4o to generate interview questions, which show that minimally-guided LLM-generated questions are no better or worse than human-authored questions with respect to clarity, relevancy and informativeness, and that LLM-generated questions outperform human-authored questions in mistake-guided question generation. The outcome of the thesis is to shed light on how embeddings-assisted techniques can be integrated into existing requirement extraction and elicitation practices to enhance and enrich them, while advancing our understanding about how stakeholder requirements may be elicited more effectively and comprehensively.

Acknowledgments

I would like to express my sincere gratitude to my amazing advisor, Travis Breaux, for the invaluable guidance and unwavering support throughout my Ph.D. research. I began working with him even before he became my advisor during the summer before I officially started my Ph.D., introduced me to the field of requirements engineering, showed me how the field could interact with other fields such as natural language processing and machine learning, and guided me on various papers I ended up submitting during my Ph.D. journey. This could not have happened without all his help and support.

I also would like to express my deepest appreciation to the other members of my thesis committee, Christian Kästner, Bogdan Vasilescu, and Fabiano Dalpiaz, for their kind and professional guidance on my thesis. Their valuable feedback and discussions with me helped me improve my thesis significantly and allowed me to improve my work in general.

Furthermore, I would like to thank my research collaborators for the insightful discussions and valuable help during my research. I would like to especially thank Anmol Singhal, not only for collaborating with me on the work described in Chapter 6, but also for all the brainstorming and fun chats with me, as well as thank Devdatta Mahesh Joshi, for collaborating and helping me on the realization of the methods described in Chapter 5. I would also like to thank Chenyang Yang and Sarah Santos, for all the interesting chats on research and on other fun topics throughout these years. For this, I would sincerely thank Carnegie Mellon University for admitting me as a Ph.D. student and allowing me to work with all these amazing people.

In addition, I would like to thank my wonderful friends, Ke Wu and Bingkai Wang, for every Saturday night we spent together and the various trips we went together, all of which are so full of fun. I would also like to thank my many Cornell undergraduate friends, Yixin Shi, Yiteng Guo, Danqiao Yu, Zhelun Zhang, among others, whom I keep meeting regularly on trips, for all the joy they brought to me. For this, I would also sincerely thank Cornell University for admitting me as an undergraduate student, helping me build my technical knowledge foundation, and bringing me all these cool friends.

Lastly, but most importantly, I would like to extend my sincere thanks to my husband Shengyuan Hu, and my parents, Zhong Shen and Heng Chen, for their unwavering love and endless encouragement throughout this journey. I met Shengyuan at Cornell where we were both undergraduate students, and luckily continued our journey together at CMU, both as Ph.D. students. There were so many times where I brainstormed and discussed research ideas with him, got muses from him, not to say the majority outside of research times, the joy of having someone who is always with me, supporting me unwaveringly no matter what I am doing. My parents, although not physically in the United States, also offer their endless support to me on all aspects of my work and life. Thank you all, forever.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Requirements Engineering (RE) faces fundamental challenges in extracting and eliciting stakeholder requirements. Stakeholders are people who have a stake in the software system, including project managers, customers, and users [183]. Stakeholder requirements are often tacit, specialized, hard to express, and require significant effort and high-level expertise to extract successfully. Traditional requirements elicitation techniques, although generally effective in extracting and formulating stakeholder requirements, often struggle with domain knowledge gaps between analysts and stakeholders, ineffective communications, and the difficulty of systematic capture of stakeholder requirements from textual information or conversation-based sources. In requirements engineering, requirements may be categorized as *functional*, meaning the system must provide functionality to satisfy the requirements, and *non-functional*, which often covers behavioral or cross-cutting concerns, including quality requirements [72]. The challenges described above are particularly prominent in domains where, apart from functional requirements, non-functional requirements like stakeholder preferences play a very important role, yet are diverse, personal, and not easy to capture. Preferences are an important aspect towards software personalization, which 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 [59]. Improving extraction and elicitation effectiveness can help produce software that is more personalized and tailored to the needs of their users.

This thesis focuses on studying the extraction and elicitation of stakeholder preferences and needs. An example target we focus on studying is *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 the software that provides these services is 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. However, despite the popularity of directory services, many of the user preferences regarding them have largely been ignored or overlooked in both scientific research and state-of-the-art practices, which is a gap we demonstrate in this thesis and we aim to bridge. 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 more than 1.6 billion diners worldwide [142]. In contrast, in 2021 the AllTrails.com web app serves a specialized hiking community of more than 25 million registered users by indexing over 200,000 trail guides in 190 different countries [149]. 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.

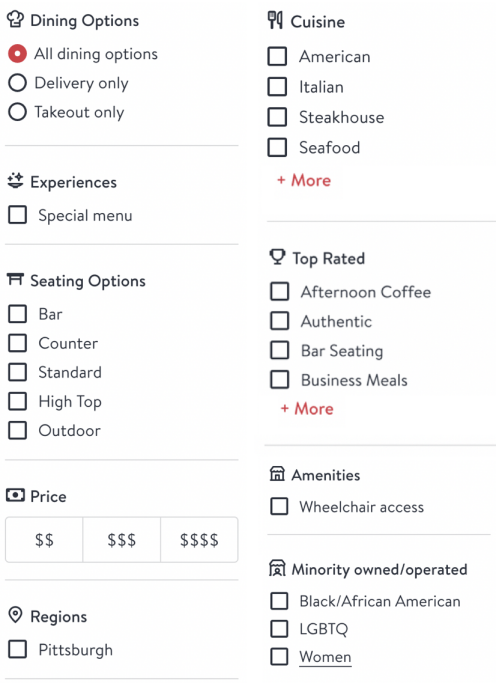

Figure 1.1: Example Preferences in OpenTable App

Figure 1.1 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 requirements satisfied by filters in the OpenTable app. The preferences indicates stakeholder needs that go beyond basic functional requirements, yet they are crucial for creating software that truly meets user expectations. As designers of directory services increase their knowledge of user preferences, they can further tailor these services to increase 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*. In requirements engineering, requirement elicitation is the process of seeking, uncovering, acquiring, and elaborating requirements for computer-based systems [202]. Requirement analysts use various techniques to elicit stakeholder needs, and these

needs can be used to anticipate opportunities to improve software and to develop custom features that allow applications to stand out from their competition. Where needs are tacit, the elicitation process must be able to draw out these needs. The challenges lie not just in identifying these needs and preferences, but in systematically extracting and modeling them from the rich natural language expressions that stakeholders use to describe their needs. For example, when stakeholders are asked to write scenarios to express how they look for a future apartment, they embed their preferences about many aspects, such as safety, commute distance, neighborhood characteristics, and accessibility, and often in tacit or indirect ways. Similarly, during requirements elicitation interviews, which are a common technique used to elicit stakeholder requirements, preference information may be overlooked due to insufficient domain knowledge, heavy cognitive load, or communication barriers, among many factors. A lot of practices of preference extraction are still largely based on statistical models of user-system interaction, with little documentation of what users prefer when they evaluate information online. In this thesis, we aim to uncover methods to ease the difficulty associated with direct elicitation of preference from users instead of relying solely on statistical models that indirectly monitor the preferences, and thus to improve the practice of direct elicitation.

Automated techniques to support elicitation and text analysis are often knowledge agnostic, meaning the techniques are limited to the knowledge that is presented to them by user. With advances in machine learning (ML) and Natural Language Processing (NLP), however, new opportunities exist to improve and automate preference elicitation tasks. In particular, human access to domain knowledge is challenging in that such knowledge is often tacit and specialized. Requirements elicitation practices often see imbalances in domain knowledge between requirement analysts and stakeholders. Accurate and comprehensive domain knowledge is important in that it serves as a foundation for defining and validating software requirements by ensuring that the software aligns with the real-world needs of the domain, and imbalances in domain knowledge may cause incorrect definition or failure to validate stakeholder requirements. Such imbalances may be adjusted by utilizing natural language embeddings, via 1) domain knowledge modeling: identify and obtain requirement-related domain elements; and 2) guided elicitation: given requirement-related domain elements, use them to refine requirement artifacts such as user-authored scenarios, or interviews. For example, word embeddings, trained on natural language text, encode semantic relationships between concepts within the domains they are being trained on, while masked language models can predict missing relationships in domain knowledge that are otherwise hard to observe. Large language models have also demonstrated remarkable capabilities in understanding and generating natural language, supporting elicitation guidance. However, a significant gap exists between these technological capabilities and their practical application in requirements engineering. In this thesis, we aim to address the gap by studying systematic methods for leveraging this information to enhance domain knowledge modeling, and improve requirements extraction and elicitation practices.

Hence, my thesis statement is the following: ***Natural language embeddings, which encode semantic information, can be used to support requirements acquisition by supplementing gaps in stakeholder knowledge and using this knowledge to guide elicitation.***

To evaluate the thesis statement, we focus on the following embeddings-assisted methods for

requirements extraction and elicitation: 1) we study the efficacy of domain knowledge modeling with user-authored scenarios; 2) we explore domain knowledge modeling with masked language models (MLM); 3) we work on refining requirement artifacts by guiding preference elicitation practices in interviews with domain knowledge obtained from the MLM; and 4) we describe methods to generate requirements elicitation interview questions with an instruction-tuned LLM, which may be used to refine and support elicitation interviews.

With the study of each of these methods, we hope to shed light on how embeddings-assisted techniques can help improve requirements extraction and elicitation. This thesis makes several key contributions to requirements engineering research and practice. First, we demonstrate an end-to-end automated pipeline for extracting stakeholder preferences from user-authored scenarios. Second, we introduce novel methods for bootstrapping domain knowledge modeling using masked language models, enabling extraction without requiring a pre-existing scenario corpus. Third, we describe a real-time machine-guided elicitation method that suggests domain elements during interviews. Fourth, we establish a framework of common interviewer mistake types, and demonstrate that LLM-generated questions can match or exceed human performance in avoiding these mistakes and are better at question relevancy, clarity, and informativeness. These contributions collectively advance the state-of-the-art in automated requirements acquisition, and demonstrate how embeddings-assisted techniques can enhance traditional requirements acquisition practices by supplementing analysts with domain knowledge, guiding elicitation processes, and helping uncover stakeholder preferences that might otherwise remain tacit or unexpressed.

The rest of the thesis is organized as follows: in Chapter 2, we present background and related work; in Chapter 3, we present our research on stakeholder preference extraction from user-authored scenarios; in Chapter 4, we present our research on domain knowledge modeling from word embeddings; in Chapter 5, we describe our research on guided preference elicitation in interviews; in Chapter 6, we describe our work on requirements elicitation question generation; in Chapter 7 we outline the thesis's main contributions, talk about how this thesis is tied into the larger context of requirements engineering, and finally conclude the thesis.

Chapter 2

Background and Related Work

2.1 Goals, Desires and Preferences

In requirements engineering, stakeholders are people who have a stake in the system, including project managers, customers, and users [183]. 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. There are *functional* requirements, meaning the system must provide functionality to satisfy the requirements, and *non-functional* requirements, which often covers behavioral or cross-cutting concerns, including quality requirements [72]. Jackson defines requirements as optative statements about the way that stakeholders want the world to be [86]. Requirements can be expressed as *high- and low-level goals* to be achieved or maintained by the system [50], and this includes system qualities or *soft-goals* [135], such as privacy, safety, etc. Desirable and necessary goals have been called optional and mandatory, respectively [112]. In this case, a desirable (or optional) goal is one that a developer may choose to implement or ignore. Similarly, preferences, which include stakeholder desires, can be understood by developers to be optional. However, preferences can be indicators of opportunities to innovate and better meet stakeholder needs, and stakeholders are understood to often prioritize over the preferences, when addressed with a particular situation in a given context, as Liaskos et. al. have described in their work of preference modeling [116].

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 [133]. Stakeholder preferences are expressed as priorities over high-level qualities, or as desired operational details [112]. 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. Preferences are non-functional requirements, because they express a stakeholder's desire for a quality or attribute monitored by a system [72]. In Jackson's view of requirements engineering [86], 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. As Jureta et al. have described in their revised core ontology for RE based on Zave and Jackson's formulation, stakeholders have basic concerns that cover beliefs, desires, intentions, and attitudes, which must be incorporated into the RE ontology to formulate the 'requirements problem' and define what it means to successfully complete RE [91]. Today in software engineering practices, 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 the software 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. One aim of this thesis is to explore methods to extract and elicit stakeholder preferences so that software developers could more easily maximize them, to make more personalized software with richer environment models.

2.2 Software Personalization with Requirements

Personalization began in product manufacturing in the 1980's, when Stanley Davis [43] 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 [106]. In the last two decades, the shift toward personalization in information systems has outpaced manufacturing, as software developers combine agile methods [27], A/B or split testing [99], real-time user telemetry, and automatic updates to rapidly optimize software and increase user dwell time and engagement. 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 [148]. Automated techniques for one-to-one marketing rely on user modeling [59], 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 [169]. 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. On the other hand, stakeholder preferences can be directly elicited from stakeholders by asking them to articulate their preferences through different approaches and in different situations. 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. An aim of this thesis is to uncover methods to ease the difficulty associated with, and to improve the practice of direct elicitation of preferences from stakeholders, guided by embeddings-assisted methods.

2.3 Crowd-based Requirements Engineering

Crowd-based requirements engineering (CrowdRE) is an emerging approach within the broader field of requirements engineering (RE) that leverages the collective input of a large, diverse group of stakeholders that are usually current or potential users of a software product, as opposed to traditional RE practices that typically involve gathering requirements from a limited set of representatives, in interviews or focus groups [76]. CrowdRE utilizes the collective wisdom of the crowd to gather, refine, and prioritize requirements, complementing traditional RE methods to provide a more comprehensive and potentially more accurate view of validated stakeholder requirements.

CrowdRE strives to mobilize as many crowd members as possible to gather requirements, hence usually cannot avoid obtaining large amount of data, oftentimes in natural language, from the crowd. Human effort is valuable and we want to expend it only when required, thus automated approaches are key to this field [76, 125]. Common CrowdRE approaches to gather requirements can involve pull feedback [127], which means software company explicitly asks the crowd for feedback or implicitly gathers feedback from analyzing legacy documents; push feedback [127], when users actively or implicitly request enhancements and changes or report issues, pains, needs and requirements in emails, forums, ticket or groupware systems; sequential task design [124], when crowd workers are asked to generate ideas in two sequential phases, and so on. When analyzing feedback gathered from different channels, automated techniques such as text mining [127, 131], speech-act-based analysis [132], NLP classification, clustering and categorization methods [100], social mining and domain modeling [192] are often used. For example, AI based techniques have been used on crowd-generated datasets to predict ratings [167], generate domain-specific glossary terms [67], observe the effects of requirements on sustainability [159], etc.

A lot of challenges still remain for CrowdRE practices, including motivating crowd members to participate, the actual elicitation process to gather useful feedback, and data challenges - the analysis of collected data to elicit requirements [76, 125]. This thesis aims to enhance CrowdRE elicitation practices, by using guided elicitation to ease the feedback gathering process, and automated methods to analyze collected data to extract preferences and requirements.

2.4 Domain knowledge Modeling in Requirements Extraction

Requirements extraction and elicitation has been the subject of significant study [60, 109, 146]. Requirements can be acquired using various techniques, such as: analyzing the direct and indirect system stakeholders to determine stakeholder needs and generate user stories [49]; combining object-oriented system analysis [77, 138], task analysis and prototyping to better account for end-users and the usage context [177]; gathering informal text descriptions of user requests with crowd sourcing methods, and automatically classifying those requests into requirements classes based on keywords [110], using a capability-based framework to unify ML model specifications for better ML engineering [33], among others. Techniques can also be used to enhance creativity during the elicitation of user stories [124]. 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 [112, 116].

As computing becomes increasingly pervasive and ubiquitous, software's ability to observe the world through user feedback, reviews and wearable device telemetry is also increasing. This allows software, and specifically directory services, to guide users toward satisfying requirements that are not controllable by software. For example, a user of a restaurant finding app can use the app to find a restaurant with the right atmosphere and cuisine due to the observability of these qualities presented in user reviews, while the software does not control these qualities. This class of software is called *directory services* and this software satisfies a class of non-functional requirement that we call *stakeholder preferences*. The challenge for developers of this software is how to identify these preferences to ensure that their services have observability into which objects and spaces described in the directory satisfy these requirements, more or less. Advances in Machine Learning (ML) and Natural Language Processing (NLP) have provided endless opportunities to improve and automate various preference elicitation processes. We aim to attempt at addressing such challenges, to use embeddings-assisted methods to aid in identifying and extracting these non-functional requirements.

Domain models capture requirements for systems. Accurate and comprehensive modeling of domain knowledge serves as a foundation for defining and validating software requirements by ensuring that the software aligns with the real-world needs of the domain. Requirements and software components may be identified via careful domain analysis that creates structured domain models [176], often from natural language domain specifications or expressions [170].

Techniques exist for automatically extracting domain model elements, such as using rules based on information retrieval, using natural language dependency parsing [12], and so on.

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 [129]. Dependencies have been used in requirements engineering to extract assertions from NL specifications for use in formal verification [166], legal meta-data from regulations [175], and software features from user manuals [151]. They have also been used to classify requirements as either functional or non-functional [45, 98], 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) [155]. 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, 45, 98], more work is needed to generalize these approaches. Typed dependencies may be able to assist in domain model construction. Whereas typed dependencies describe binary, grammatical relations among words, phrase structure grammars relate words to nested phrases that are typed based on their grammatical role, such as a noun, verb, or prepositional phrase [66]. Arora et al. combine phrase structure grammars, typed dependency parsing, co-reference resolution and stop words with 18 domain model extraction rules to identify concepts, associations, cardinalities and attributes in four industrial natural language requirements documents [12]. This approach has been extended using active learning to reduce false positives by 45% to yield a .96 precision [15]. Saini et al. combine named entity

recognition and co-reference resolution with rules to extract classes, relationships, attributes and cardinalities from problem descriptions [171].

Other ML models may also assist in domain knowledge modeling. For example, an API-based method was used to categorize software into domain categories [115], a LLM-based zero-shot method generates knowledge bases to support systematic exploration of domain concepts to guide model testing and support requirements elicitation [199], and a LLM-based method semi-automates domain model derivation from transcripts of requirements elicitation conversations [137]. Yet until today, especially since LLMs are still a relatively new tool, a lot are left to be explored about utilizing semantic information that are naturally embedded in ML, particularly embedding-based models, to aid requirements extraction and elicitation. A part of this thesis focuses on generalizing typed dependency techniques to create domain models for directory service software, while we also aim to explore a wider range of embeddings-assisted methods and natural language techniques, involving MLMs and LLMs, to model domain knowledge, and bridge the gap between the new technologies and traditional requirements extraction practices.

2.5 Requirements from Texts

Techniques to extract requirements from texts have been used to identify legal primitives in laws [93], mine software requirements in app reviews [9,90,182], extract feature models [3,5,172] and to perform domain modeling [12]. Legal compliance checking can be automated through the extraction of compliance requirements from legal documents [93]. 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 [90]. Acher et. al. describe a pipeline for extracting feature models from product descriptions [5]. Domain models that describe knowledge about domains [26] 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 [12].

Sleimi et al. describe a tool to automatically extract requirements information from legal texts using a concept typology [161]. The typology includes statement-level types (e.g., definitions, permissions and obligations), and phrase-level types (e.g., actor, action, modality, time) can be used to extract text-based concept instances from text.

Typed Dependencies, as mentioned above, is one potentially viable method to extract requirements. Another method that has been used to extract requirements from text is Named Entity Recognition (NER) [68,81]. 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 [108]. Entity typing aims to classify entities into a larger typology, upwards of more than 10,000 types. Dai et al. combined Hearst patterns with a masked language model to generate weak labels for entity typing [52]. 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 [181]. 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 [32].

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) [41], conditional random fields (CRFs) [114], and neural network models, such as bidirectional long-short term memory (Bi-LSTM) [83], convolutional neural nets (CNNs) [40], and transformers [185]. 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% [195]. A limitation is the need for large training corpora, which is being addressed by emerging unsupervised and semi-supervised ML techniques [139].

Whereas NER is used to classify entities into a shallow taxonomy, methods exist to identify richer ontologies for entities. Youn et al. employed word embeddings to extend a food ontology from an ontology scaffold [198]. The extension is built using a mapping function that maps word embedding vectors to a target class, and the performance is evaluated using measures of granularity and cohesiveness. Wang and He identify hypernyms using bidirectional residual relation embeddings (BiRRE) to create a latent projection model with negative regularization [189]. In this model, a candidate hypernym-hyponym term pair is represented as a BiRRE vector, and project model is used to simulate the generation of these vectors. Ravichander et al., inspired by the Cloze task, describe using MLMs to discover hypernyms, which are more general categories for words [22]. The MLM is trained by randomly masking some of the tokens from the natural language inputs, and aiming to predict the original vocabulary id of the masked word based only on its context [44]. Since words from both sides of the mask are used in training the model, the model is bi-directional in nature. At inference time, the model is usually given a previously unseen natural language input with a [MASK] token in it, and asked to predict the most likely substitutes for the masked token. The model would usually give possible substitutes to fill in the mask, and each substitute's confidence score, a value between 0 and 1 indicating how certain the model is that the substitute is correctly assigned, with a confidence score of 1 indicating certainty. An example masked input can look like "I watched a [MASK] at the cinema and it was fun", where the model tries to predict the best work to fill in the [MASK]. MLM is a very useful part in training a language model in an unsupervised fashion, because it helps the language model understand relationship between tokens. Such language models can then be fine-tuned to adapt to specific downstream tasks. A part of this thesis studies the efficacy of using transformers-based MLMs for domain knowledge modeling and requirements artifact refinement. Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based language representation model that utilizes the attention mechanism to learn word embeddings. BERT is pretrained with two unsupervised tasks, a MLM and a Next Sentence Prediction model, to understand relations between tokens and relations between sentences. The parameters of BERT can then be fine-tuned to adapt to specific downstream tasks [185].

Recent appearances of large language models (LLMs), such as the Generative Pre-trained Transformer (GPT) models, are large scale language models based on the Transformer architecture with improved ability of understanding and producing text. In particular, autoregressive training techniques for building large language models (LLMs) have improved the capability of understanding and generating natural language texts [24]. Whereas more traditional state-of-the-art models

like BERT are bidirectional transformers trained with separate tasks (a masked language modeling task and a next sentence prediction task) in order to teach them to understand natural language syntactically and semantically, autoregressive models like GPTs are large scale models, unidirectionally trained to predict each upcoming word in the text given all previous words, which leads to improved ability to generate coherent and contextually appropriate text as opposed to just natural language understanding. Due to the improvements of LLMs on natural language generation, the latter provides more flexibility for open-ended tasks where the length and structure of the output are not predetermined. LLMs have been explored and shown to be successful in various requirements engineering tasks, such as use case model extraction [21], goal modeling [35], among others. Yet, there are still a lot left to be explored until we understand the full potential LLMs can achieve in aiding requirements practices. This thesis contributes to this ongoing research area by studying the efficacy of LLMs in improving certain aspects of elicitation practices.

2.6 Requirements from Interviews

Interviews are one of the most traditional and commonly used requirements elicitation technique, which provides a systematic and interactive way for requirements analysts to directly engage with stakeholders and ask them questions to understand their needs, preferences and expectations regarding a service [79, 145, 164, 202]. While text-based preference elicitation permits users to explore and propose entities they prefer by writing text, interview elicitation practice engages users in direct conversations and allows the interviewer and stakeholder to explore their preferences in a dialogue. Unlike scenario authoring or other forms of text-based elicitation methods, dialogue provides the interviewer the opportunity to direct the stakeholder's speech.

Interviews, as a kind of inquisitive elicitation technique, provide many advantages. Not only do interviews serve as a direct form of conversational engagement with potential stakeholders, they also allow interviewers to delve deep into the problem domain, gain insight to stakeholder preferences, and resolve possible ambiguities around multi-faceted, hard to express or even conflicting viewpoints [202]. In interviews, requirements are co-created by the participating requirements analyst and stakeholder [62]. Interviews may be conducted in a one-on-one format, wherein an interviewer converses directly with a stakeholder, or within group, wherein multiple participants or even analysts all engage in the conversation session. Group interviews with mediators are typically called focus groups [174]. Interviews may be structured, where a pre-determined list of questions are prepared and asked during the interview; unstructured, with no advance preparation of a script or questions; or semi-structured, where interviewers adapt their questions to the conversation based on interviewee responses during the interview time [102, 202]. Structured interviews are more rigorous, but can limit the investigation of new ideas, while unstructured interviews may have the risk of completely neglecting some topics, focusing in too much detail in some areas, and not enough in others [202]. Interview questions may either be closed-ended, requiring a direct, coded response, or open-ended, where the interviewee is encouraged to provide detailed, exploratory answers that reveal insights into needs, expectations, and potential issues [191].

Albeit the variety of interview formats, conducting interviews all involve a lot of challenges. Stakeholder preferences are often tacit knowledge that require significant effort to draw out ex-

plicitly [63]. In particular, knowledge experts may become more skilled but less aware of the cognitive process involved, and also due to limited storage capacity of attentional memory, knowledge elicited is limited to that which is readily available to the conscious introspection by the expert at the time of inquiry [16]. Interviewers need specific skills, such as interpersonal skills to create rapport with interviewees [30], and a decent amount of domain knowledge and subject-matter expertise on the interview topic, in order to ask relevant, meaningful, and thought-provoking questions [30], understand interviewee responses, draw out tacit knowledge [63] and identify gaps and opportunities to follow up and dig deeper for further elicitation [82]. Cultural and linguistic barriers may also hinder communications [62, 202], as communications are essential in interviews and a lack of sufficient language proficiency or mutual cultural understanding would devastate the elicitation. Some interviewers may suffer from anxiety or self-doubt, rendering the overall interview experience less engaging or comfortable. Besides, in psychology, it has also been shown that some aspects of human nature can affect interview quality. Interviewers need to consciously avoid tunneling, which is when interviews talk excessively about one topic and fail to cover the necessary breath of interview topics [53]. Interviewers also need to suppress jumping between different topics, which may result in less cohesive experiences and more superficial requirements [29]. The questions asked by the interviewer during an interview are also very important. A lack of domain knowledge may hinder asking appropriate follow-up questions during the interview [82]. Asking an excessive amount of irrelevant, superficial or off-topic questions or forgetting to come back to a previously discussed line of inquiry can reduce elicitation quality [31]. Since interviews are conducted in real time, interviewers may need to combat with the stress associated with having to come up with appropriate follow-up questions in real time, which would be devastating, since these questions can clarify ambiguity, validate interviewee statements, and dive deeper into a mentioned topic, identifying opportunities for more detailed insights [17]. Such stress may stem from an excessive cognitive load, built during interview due to having to communicate back and forth with interviewees and at the same time process interviewee responses and formulate follow-up questions [78], or from information overload, where interviewees provide long or complicated speech that may contain multiple points of interest that is hard to process [14]. The misinterpretations and cognitive limitations during communications could hinder the use of interviews for elicitation. This thesis attempts to use embeddings-assisted, natural language processing techniques to tackle some of the above challenges, including unfamiliarity with domain knowledge, difficulty coming up with follow up questions in real time, failure to appropriately probe into topics of interest, information overload, and tunneling and forgetting to come back to other previously mentioned topics that could hurt elicitation coverage.

Many existing literature has described various criteria for practitioners on the appropriate conduction of elicitation interviews, involving criteria on the appropriate opening and closing of interviews, the right ambience and flow, the question framing, question contents, the proper elicitation goals, and pointed to common mistakes made during elicitation interviews that are for practitioners to avoid when they interview stakeholders. For example, interviewers may open or close the interview in a wrong way, asking for ideas without giving appropriate guidelines [30, 48], failing to build rapport before asking questions [30], or not providing a summary before the end [30, 48]. Ambiguities may not be properly leveraged, including tacit assumptions [150] or tacit knowledge

known to stakeholders but not to the analysts [30, 48] not properly elicited, unclear or contradictory speech made by stakeholders not properly addressed [30, 48], alternatives not properly considered [30, 150]. Questions may be too generic or domain-independent, or understood incorrectly [30]. Implicit stakeholder goals may not be explicitly stated [48], and implicit stakeholders may not be identified [30, 30]. Feature priority [30] and resource limits [48] may not be clearly elicited. Interviewers may go back and forth among topics leading to confusion, or run out of questions because of an interrogatory-like style [48]. Questions may not be properly framed, such as questions too long or articulated [30, 48], too vague, technical, or irrelevant [30], inappropriate to stakeholder's profile [30], involve multiple kinds of requirements [48], ask for solutions [30], contain jargon [30], among others. Moreover, interviews should not be unethical or disrespectful [84]. The thesis includes a framework of common interviewer mistake based on past literature, and discusses interview question generation techniques to generate interviewer questions that avoid some of the mistakes in the framework.

Since interviews naturally involve a lot of natural language, automated techniques involving the use of Natural Language Processing (NLP) have provided many opportunities to improve the elicitation process. The interview, as a form of conversation, involves a large amount of natural language text that often disregards grammatical rules of written text, but which can still be analyzed using NLP pipelines. The Masked Language Model (MLM) [185], mentioned above, can be one such component in a pipeline. In addition to MLMs, other techniques that are useful in analyzing stakeholder speech include automated transcription and part-of-speech tagging. Automated audio transcription [71, 194] is a speech recognition technique that converts spoken language into text format, and can be done in real-time or after audio collection, the former being more challenging, as it demands the rendering of written representation of spoken words as they are being spoken, in real-time. Part-of-speech tagging [160] assigns grammatical categories to a designated word in a sentence based on its syntactic function. These tags can be used to identify adjectives that describe nouns and can also describe the qualities of things preferred by stakeholders. Autoregressive training techniques for building LLMs, like the GPT models mentioned previously, have the advantage of generating coherent, high quality natural language text, particularly for open-ended tasks. Interviews, as an important requirements artifact, is an example of an open-ended task that may benefit from LLMs. For example, LLMs are used to generate simulated users for interviews and analysis to elicit requirements [4]; to generate requirements elicitation interview scripts [64, 65]; and to post-process interview transcripts [168]. Moreover, some literature attempted at building chatbots that are effective at engaging users and eliciting quality information, but found challenges such as the lack of clarity for generated interview questions, and chatbot-unrecognized user input [84].

Despite this prior research that is mainly focused on preparing and post-processing interviews, research to assist the elicitation process during interview time is relatively limited. In this thesis, an important part of the work aims to bridge this gap by describing an LLM-assisted method to guide follow-up question generation that may be used to support requirements elicitation interviewers.

Chapter 3

Domain Knowledge Modeling using User-Authored Scenarios

Human access to domain knowledge is challenging in that such knowledge is often tacit and specialized. To extract and obtain domain knowledge from natural language text-based contents sorely by human experts is both time-consuming and costs a lot of manual efforts. On the other hand, advances in Natural Language Processing has provided a potential for using embeddings-assisted techniques to automatically identify and obtain requirement-related domain elements from natural language contents, and in turn facilitates and enhances domain knowledge modeling. In this chapter, we study how domain knowledge could be modeled by applying such techniques to user-authored scenarios.

3.1 Motivation and Goal

In agile software development, developers are frequently experiencing pressure to deliver increments of working systems that broadly satisfy a wide variety of users. Agile development has led to limited documentation (e.g., replacing use cases with user stories), and this approach can overlook the more nuanced stakeholder preferences that could lead to improved personalization. We recognize that developers need automated methods to extract requirements to meet the demands of personalization in the agile environment, to obtain requirements more efficiently and without losing the more nuanced preferences. 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. Our research in this section 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; (4) Finally, 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.

3.2 Research Questions and Methodologies

Our research in this section 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 or techniques are best suited to automatically extract preferences from scenarios?

RQ1 and **RQ2** aim to understand preference types and categories in user-authored scenarios and how much the existing directory services capture or overlook these preferences, to motivate why automated preference extraction is important and how it is viable via scenario extraction, while **RQ3** studies the actual viability of automated preference extraction from scenarios.

Our approach consists of the following steps (see Figure 3.1), 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 3.1, 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.

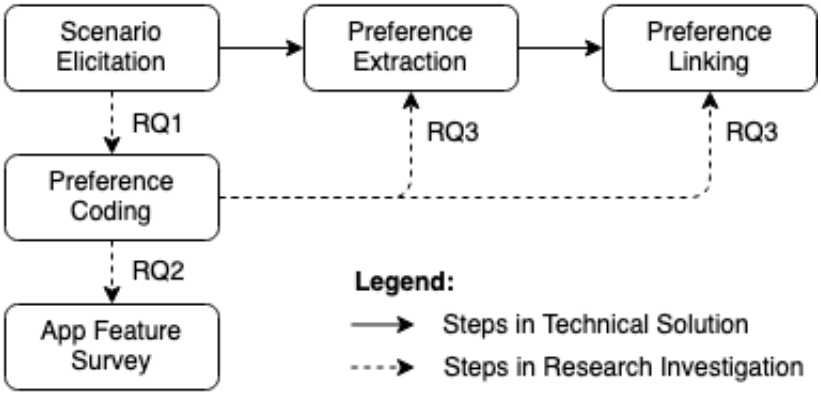

Figure 3.1: Research Method Overview

3.2.1 Scenario Elicitation

To obtain the necessary data to answer RQ1 and RQ2, and to provide a dataset for RQ3, a 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 collected an English-language, user-authored scenario corpus that contained scenarios for 12 selected prompts. This yields a total of 207 scenarios. The prompts used in this step appear in Table 3.1. 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.

The scenarios were collected using Amazon Mechanical Turk in an IRB-approved research study. Workers 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

Table 3.1: 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?

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.

This scenario corpus serves as the foundation of all our research methods. We analyze the scenarios via preference coding to understand how stakeholders describe their preferences to answer RQ1, get the described preferences to evaluate existing directory services to answer RQ2, and use the corpus together with the obtained preferences as the ground truth dataset to automatically extract preferences in order to answer RQ3. The details of how we use the corpus are shown below.

3.2.2 Preference Coding

Grounded theories are "grounded" in the data, which can yield strong evidence to support the theory, but which also limits generalizability [37]. In this chapter, 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 [157].

To answer RQ1, how do stakeholders describe their individual preferences in scenarios they author, we coded the scenarios using a three-cycle coding process to produce the grounded theory [157]. 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" [157] 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 [157] 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, two of our researchers separately performed a third cycle using theoretical or "closed coding" [157] by applying the coding frame to relevant words in a new sample of 20 scenarios. This step is down in preparation of answering RQ3 later, by creating a ground truth dataset of preference phrases. 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 [111]. After reviewing and resolving differences in the sample, updating the coding heuristics, and re-coding the same sample, both researchers achieved a 0.94 Kappa, which shows high above-chance agreement acceptable in computational linguistics [10].

Figure 3.2 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 3.3.

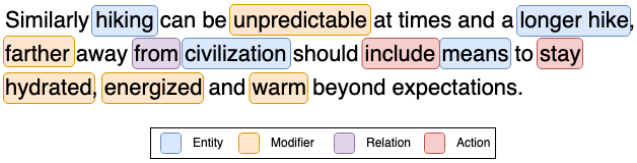

Figure 3.2: Example Sentence Coded with Coding Frame

The preference coding step helps discover how stakeholders describe their individual preferences in scenarios they author by extracting and encoding the preference-related domain knowledge elements. With these preference elements, we then could go on to evaluate how existing directory services may or may not satisfy the described preferences, and use them as the ground truth for our automated preference extraction methods.

3.2.3 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 3.2 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 chapter to indicate labeled words and phrases.

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

Table 3.2: 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

second authors coded the web apps using the feature categories [157], and the results are presented in Section 3.3.1.

3.2.4 Preference Extraction

We sought to answer RQ3, what natural language features are best suited to automatically extract preferences from scenarios, by first using a technique: we use a typed dependency parser to identify nouns that were the subject of verbs in scenarios. This method is a naive technique to build a domain model that is preference free, and limited to nouns and verbs. Of course, these nouns and verbs may imply preferences because the authors chose them, but we also want to look for more explicit preferences, not just implied preferences. Hence as a next step, we extend the above technique to 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. These approaches learn classifiers to predict preference elements from scenarios, and are not just limited to nouns and verbs. For all of these three approaches, we use the result from preference coding, as demonstrated in Section 3.2.2, 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 3.3.2. In all evaluations, 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).

Rationale for Model Choices

The naive technique used a typed dependency parser, because typed dependencies provide a scenario text representation to complement word distributions, and because they are popular in automated requirements analysis [45]. For the extensions, 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 [147], and the Stanford Named Entity Recognizer provides a reliable implementation [57]. The second model we chose is a random forest classifier trained on typed dependencies. 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 [195], we choose to apply BERT to our task of preference extraction.

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 [129]. In Figure 3.3, 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 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.

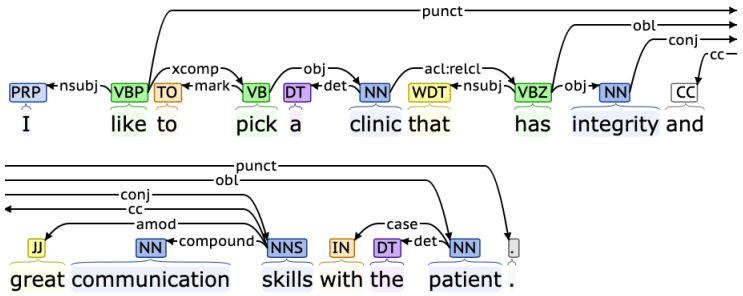

Figure 3.3: Example Typed Dependency Graph

We used typed dependency parsing to identify nouns that were the object of verbs in scenarios to extract simple domain models. For example, in the apartment finding domain, this approach yields commonly used concepts, such as "apartment" and "budget," with link from these concepts to common actions over those concepts, such as "visit," "view," and "show" for user actions on apartments and "compile," "fit," and "determine" for user actions on budgets. To that end, we used the Stanza toolkit [152] based on the CoreNLP framework to identify verbs and dependent nouns in obj or obl typed dependencies. The results are reported in Section 3.3.2.

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 3.2.2. Figure 3.4 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 3.4, 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.

```
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 3.4: Example Training Data for CRF-based Named Entity Recognizer

Random Forest

We trained a random forest classifier using the typed dependencies acquired using the Stanza library [152]. 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 2*w* - 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 3.3 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 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.

Bidirectional Encoder Representations from Transformers

The Bidirectional Encoder Representations from Transformers (BERT) [44] 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 pre-trained DistilBERT model [163], downloaded from the HuggingFace library [186], 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.

Evaluation Methods

We evaluate 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

[1]Figure 3.3 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.

20% for validation, which includes tuning hyper-parameters and choosing the number of epochs for training. We report our findings in Section 3.3.2.

3.2.5 Preference Linking

Preference extraction described in Section 3.2.4 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 [89]. 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 3.5 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 3.3.3

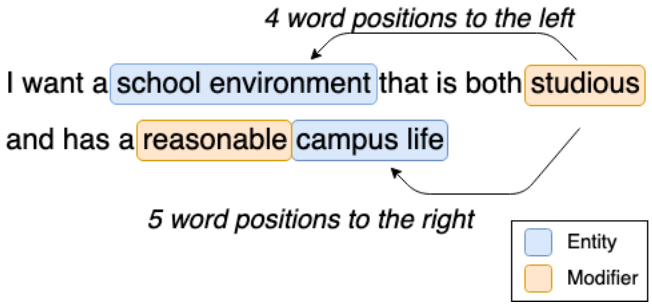

Figure 3.5: Preference Linking using Word Distance

3.3 Results

We now report the results of eliciting and analyzing scenarios for preferences.

A total of 151 distinct crowd workers responded to 12 scenario prompts to form the scenario corpus. 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 two researchers, independently annotated a sample of 20 scenarios using the coding frame described in Section 3.2.1 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 3.3 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 3.2.1. The last row describes the average frequency *per scenario* for the total number of sentences and labeled items.

Table 3.3: 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 3.3, four categories of preference words or phrases that stakeholders use to express their preferences were discovered with grounded theory: 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.

With the elicited corpus, we now present the results of the app feature survey and automated extraction and linking of preferences from scenarios.

3.3.1 Preference Gaps in Existing Systems

As described in Section 3.2.3, 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 3.4, 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 3.4: 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 3.5 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 3.5: 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 3.6 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 3.6: 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 3.7 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 3.7: 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 3.8 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 3.4-3.8 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. We hope this result could provide insight to software developers about how their software may have ignored some important stakeholder preferences, and choose to improve their coverage where they see fit.

Table 3.8: 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

3.3.2 Automated Preference Extraction

We first present results of the naive technique, which uses a typed dependency parser to identify nouns that were subjects of verbs in sceanrios in order to build a domain model. Table 3.9 - 3.12 present the most frequently used nouns across the elicited scenarios in the apartment finding, restaurant finding, hiking trail finding and health clinic finding domains, including the number of dependencies in which the nouns occurred (**Dep.**), the number of scenarios in which the nouns occurred (**Scen.**), and example verbs where the nouns were in the *obj* or *obl* typed dependencies. Notably, the top nouns correspond to key activities in this domain: finding an apartment involves make lists, searching an area, scheduling appointments, and compiling and fitting a budget.

Noun.	**Dep.**	**Scen.**	**Example Verbs**
apartment	77	20	find, move, view, visit, sort
list	19	9	make, create, narrow, compile, rank
area	15	10	search, consider, look, investigate
place	13	10	move, narrow, find, qualify, look
appointment	8	8	schedule, make, set
budget	8	6	compile, develop, fit, determine
search	6	5	make, conduct, limit, continue
price	6	5	negotiate, sort, filter, ask
rent	5	5	pay, afford, look, split
one	5	3	call, find, choose, make

Table 3.9: Top-10 Nouns Ranked by Dependency Count for Apartment Finding Scenario

Noun.	Dep.	Scen.	Example Verbs
restaurant	39	17	find, going, choosing, brainstorm, search
food	13	7	eat, handle, acquire, deliver, like
review	12	8	check, has, use, look, go
meal	10	7	makes, take, enjoy, order, choose
time	8	6	takes, go, wait, have, make
place	7	5	looking, pick, find, going, enjoy
type	7	4	think, agree, like, have, going
something	5	5	grabbing, try, look, type, feel
option	5	5	limits, vote, has, get, look
search	5	4	narrow, do, widen, run

Table 3.10: Top-10 Nouns Ranked by Dependency Count for Restaurant Finding Scenario

Noun.	Dep.	Scen.	Example Verbs
hike	40	16	start, prefer, plan, choose, join
trail	33	12	find, search, finish, pick, walk
park	17	8	search, find, locate, review, plan
day	8	5	plan, hike, choose, have, go
backpack	7	7	prepare, pack, put
route	7	5	take, plan, find, know, prefer
water	6	6	get, pack, put, bring
experience	6	5	get, value, want, make, enjoy
weather	6	4	check, choose, encounter, have
hiker	5	5	take, allow, is, like

Table 3.11: Top-10 Nouns Ranked by Dependency Count for Hiking Trail Finding Scenario

The problem with this approach, however, is that this is a naive technique to construct a domain model that is preference free, and are limited to nouns and verbs. To further look for more explicit preferences, 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 grammat-

Noun.	Dep.	Scen.	Example Verbs
clinic	48	17	choose, research, search, call, avoid
doctor	22	11	find, see, get, pick, call
insurance	18	11	accept, check, take
review	18	11	find, read, sort, look, check
time	12	9	set, wait, spend, get, call
appointment	12	7	set, make, attend, get, have
care	10	6	get, give, receive, take, prefer
choice	7	5	rank, narrow, filter, make, think
place	7	5	pick, search, find, get, want
website	6	6	go, dig, list, visit, find

Table 3.12: Top-10 Nouns Ranked by Dependency Count for Health Clinic Finding Scenario

ical 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 as previously explained in Section 3.2.4: a *seen evaluation* and an *unseen evaluation*. Table 3.13 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.

3.3.3 Automated Preference Linking

The preference linker, described in Section 3.2.5, was used to link a word labeled with the types *modifier*, *relation*, and *action* to an *entity* to form a preference phrase. We evaluate its accuracy 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,

Table 3.13: 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**

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 3.14 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 3.14: 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 3.15 presents examples of correctly linked preferences to illustrate the results that developers could use as outputs of the overall approach.

Table 3.15: 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

3.4 Discussion

We now discuss the error analysis, privacy-sensitive preferences, and the approach's practical application, as well as challenges associated with the approach. The error analysis aims to help us understand our models' performance better; while the privacy-sensitive preferences and the approach's practical applications and challenges help us get a better view of how our approach may be situated in a larger requirements elicitation process.

3.4.1 Extraction Error Analysis

To better understand why our model works its way, especially how the mistakes have arisen, 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 3.16 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.

Table 3.16: 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

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.

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).

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 3.4.1, 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.

Error Correction

Several classifier errors may be fixed with post-processing to improve performance. For example, 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.

Model Comparison

We observe from Table 3.13 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.

3.4.2 Linking Labels to Preferences

The preference linker yielded 131 errors out of 1323 expected preference phrases (9.9%), as presented in Table 3.14. 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 [within] 10

miles of my home in which the relation word *within* links to *providers* yet *network* is in the intermediating clause.

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.

3.4.3 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.

3.4.4 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. We acknowledge the possibility that some missing requirements illustrated in the app survey may be intentionally not implemented, yet still we believe our methods may provide more insights to supplement existing requirements to build a more complete model containing the preference-related domain knowledge.

3.5 Threats to Validity

In this section, we discuss the threats to validity, including both the threats we have addressed and the threats remaining, which aligns with the suggestions made by Yin [197]. We talk about threats we have considered before the experiments were designed, how we attempted to address the threats, and the remaining unaddressed threats or newly discovered threats while conducting our experiments.

Construct validity refers to whether we are measuring what we believe we are measuring [197]. 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 [197]. 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 [197]. 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 [121]. 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.

3.6 Summary

In this chapter, 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 [205].

Chapter 4

Domain Knowledge Modeling using Masked Language Models

The preference linker in Chapter 3 only relates other types of domain elements to entities, but there are other types of missing relationships unexplored. These relations could potentially provide useful insight to software developers about how preferences, or preference elements within a particular domain are connected to each other. For example, a "kitchen" is an important component of an "apartment", and a "fireplace" can be an ideal feature in a "kitchen". Understanding different relationships within a domain can potentially lead to better design of a software system. Hence in this chapter, we explore how we can discover the missing relationships with a Masked Language Model (MLM), and how we can further extend our discovery to identify other preference related domain elements with an MLM that utilizes embeddings. This helps us to see how semantic information encoded in natural language embeddings can be used to support requirements acquisition by supplementing gaps in stakeholder knowledge.

4.1 Motivation and Goal

In requirements engineering, the requirements analyst seeks to understand the problem world, or domain, to best identify requirements for the solution [107]. Challenges to acquiring domain knowledge include that sources of such knowledge may be difficult to access, distributed or conflicting. In addition, stakeholders may have tacit knowledge that they, or the requirements analyst, fail to recognize or mention during elicitation [173]. With regard to textual representations of domain knowledge, such as scenarios, the text may be incomplete or ambiguous [134, 140]. Domain models capture requirements for systems. Accurate and comprehensive modeling of domain knowledge serves as a foundation for defining and validating software requirements by ensuring that the software aligns with the real-world needs of the domain. Requirements and software components may be identified via careful domain analysis that creates structured domain models [176], often from natural language domain specifications or expressions [170]. Automated techniques to support elicitation and text analysis are often knowledge agnostic, meaning the techniques are limited to the knowledge that is presented to them by user. With advances in machine learning (ML)

and Natural Language Processing (NLP), however, new opportunities exist to automate elicitation and domain analysis tasks.

Neural models in particular have shown strong performance across many NLP tasks. The construction of these models requires large training dataset, up to millions to billions of data. In NLP, transfer learning, in which information is learned in one or more domains or tasks and later transferred to another domain or task, has reduced the need for large datasets [190]. This is particularly true of word embeddings, which are often constructed using unsupervised machine learning methods over large corpora [134, 156, 180].

The preference linker in Chapter 3, although accurate when linking domain elements, only relates other types of domain elements to entities, but there are still other types of missing relationships unexplored. To further explore the undiscovered, missing relationships, we may use knowledge from word embeddings. We may even further extend our discovery to identify other preference related domain elements with word embeddings. This helps us further see how semantic information encoded in natural language embeddings can be used to support requirements acquisition by supplementing gaps in stakeholder knowledge.

In this section, we explore the following approach to acquire domain models: we evaluate word embeddings to discover their efficacy for discovering domain models using limited input data, frequently one word, to bootstrap the discovery. Specifically, we employ a BERT-based Masked Language Model (MLM), which is a model initialized using a word embedding and then trained with sentences that contain one missing word. MLM models have been used to study the Cloze task [54], which is a psychological test given to humans where one sentence is provided with a missing word and the human subject is asked to present the missing word. With regard to MLMs, the model is evaluated by predicting which words are most likely to fill the missing word.

We also investigate the advantage and drawbacks of this approach compared to the domain knowledge modeling using user-authored scenarios approach, mentioned in Chapter 3, and how they can be used complementary to each other.

4.2 Research Questions and Methodologies

To study the efficacy of using word embeddings to discover domain models, we aim to answer the following research question (RQ):

RQ: Given an embedding, what kind of domain knowledge can be extracted?

To answer this question, we conduct our research with the following method: we use seed question templates that include a domain-specific noun, a seed verb and a mask, and have Masked Language Model predict the substitute for the mask in order to extract domain elements to construct domain models. Compared to domain knowledge modeling using user-authored scenarios, which was a retrospective method in that the scenario corpus must exist in order to perform knowledge modeling, this method is prospective in that, with the help of the pretrained Masked Language Model, no scenario corpus is required at the time of extraction. We talk about the approach in detail below.

4.2.1 Domain Knowledge in Masked Language Models

We investigate the **RQ** using three approaches: (1) *seed questions* that include a domain-specific noun, a seed verb and a mask; (2) *seed questions with intensifier*, such as adjectives that change the intensity of the mask; (3) *seed questions with inputs*, wherein the input can be an adjective or a verb learned from a previous result. These approaches are selected with the aim of manifesting different kinds of domain knowledge that may be extracted from an embedding. We now illustrate these three approaches.

The seed question consists of a template, including a domain-specific noun (e.g., apartment, restaurant), a verb (e.g., has, is, be) and the mask [MASKED]. Seed questions may aim to elicit modifiers attached to the noun. An example seed question would be "I want an apartment that is [MASKED]", where the Masked Language Model is asked to fill in the "[MASKED]" field with possible substitutes, such as "cheap", "clean", "spacious", etc. Depending on the model's confidence score, these fillers will be ranked from highest to lowest confidence score. Seed questions may also aim to elicit entities that are associated with the domain-specific noun. An example seed question here is "The apartment has a [MASK.]" where the Masked Language Model may fill in entities like "basement", "bathroom", etc.

Next, we investigate whether including intensifiers, such as "very" or "extremely", in the seed questions will yield in more value-laden results for the masked word than the seed question alone. For example, in the apartment-searching domain, we vary the seed question of "The apartment is [MASKED]" by adding intensifiers, such as "The apartment is very [MASKED]" and "The apartment is extremely [MASKED]".

Finally, we study whether using results from prior queries can yield better results for a seed question. For example, the query "I want an apartment that is [MASKED]" may yield the result "spacious" for the mask, which is a modifier that can then be used to construct a new query "The [MASKED] apartment was spacious" or "The spacious apartment will be [MASKED]". Such queries can yield possible action substitutes, such as "vacated", "renovated", etc. Furthermore, these action substitutes, together with the modifier, may be used in another round of query to elicit actors, such as "The [MASK] renovated the spacious apartment", where the Masked Language Model may fill in actors like "owners", "couple", etc.

All these approaches help us answer our **RQ** by showing different kinds of domain knowledge that can be extracted from word embeddings. We acknowledge that the kinds of domain knowledge extracted may not be comprehensive. The results of the above approaches will be shown in Section 4.3.

4.3 Results

We report our results from extracting domain knowledge from a masked language model using three query templates: seed question, seed question with intensifier, and seed question with inputs, as described in Section 4.2.1. In all experiments below, we used the Masked Language Model from HuggingFace, pretrained on a union of five large English corpora [119]. We chose this model because it is pretrained to capture a wide range of language uses, styles, and contexts, and

updated frequently to ensure access to the state-of-the-art pretrained model version, making it highly versatile and robust.

Each section below corresponds to a query template, in which we present the masked query followed by the results, consisting of the top five unmasked words ordered by their model confidence scores (in parentheses). Please note that the results are not comprehensive due to space limits. For more experiment results, please refer to our open sourced code at [206].

Seed Question:

includes only a domain-specific noun phrase, a verb and a mask, in this case one of apartment, hiking trail, restaurant or health clinic and the verb to-be.

Seed Question to elicit modifiers: We use the seed question templates below to elicit modifiers attached to the domain-specific noun phrase.
Query: I want an apartment that is [MASK].
Result: furnished (0.0883), nice (0.0501), beautiful (0.0384), perfect (0.0372), comfortable (0.0352)

Query: I want a hiking trail that is [MASK].
Result: paved (0.1067), easy (0.0551), safer (0.0367), accessible (0.0297), hiking (0.0261)

Query: I want a restaurant that is [MASK].
Result: nice (0.0437), perfect (0.0363), amazing (0.0291), delicious (0.0256), open (0.0227)

Query: I want a health clinic that is [MASK].
Result: open (0.0690), functional (0.0399), thriving (0.0325), affordable (0.0251), good (0.0243)

Seed Question to elicit entities: We use the seed question templates below to elicit entities associated with the domain-specific noun phrase.

Query: The apartment has a [MASK].
Result: basement (0.0977), bathroom (0.0637), restaurant (0.0594), balcony (0.0519), garage (0.0518)

Query: The hiking trail has a [MASK].
Result: waterfall (0.1667), campground (0.0807), cafe (0.0745), lighthouse (0.074), fountain (0.0376)

Query: The restaurant has a [MASK].
Result: cafe (0.3504), bar (0.1818), bakery (0.0824), restaurant (0.0523), pub (0.0227)

Query: The health clinic has a [MASK].
Result: pharmacy (0.3899), clinic (0.0685), playground (0.0501), cafeteria (0.0319), library (0.0285)

We note a few observations from the above results. (1) Model responses may be words in the query, e.g., a hiking trail that is *hiking*, which are responses that can be discarded. (2) The

extracted domain model can be enhanced by finding the binary opposites of discovered adjectives using antonyms in a dictionary, e.g., a hiking trail that is *unpaved*, *difficult*, *unsafe*, or *inaccessible* as antonyms of the above responses. (3) Some qualities are clearer or more vague than others, e.g., a restaurant that is *open*, which refers to operating hours, versus one that is *amazing*, which could refer to numerous other qualities, such as dining room ambience, food quality or taste, etc. This introduces a need to refine the meaning of vague qualities, which could require a richer source of domain knowledge beyond the masked language model.

Seed Question with Intensifiers:

includes the seed question and an adjective to change the intensity of the masked word.

Query: The apartment is [MASK].
Result: vacant (0.1076), furnished (0.0755), rented (0.0701), uninhabited (0.0270), empty (0.0216)

Query: The apartment is very [MASK].
Result: spacious (0.1084), small (0.0606), modern (0.0445), expensive (0.0414), luxurious (0.0344)

Query: The apartment is really [MASK].
Result: nice (0.1473), cute (0.0369), beautiful (0.026), creepy (0.0233), huge (0.0223)

Query: The apartment is extremely [MASK].
Result: expensive (0.1267), spacious (0.0664), cramped (0.0661), dilapidated (0.063), luxurious (0.047)

Query: The hiking trail is [MASK]..
Result: paved (0.1597), accessible (0.0948), nearby (0.0788), maintained (0.0344), blazed (0.0292)

Query: The hiking trail is very [MASK].
Result: scenic (0.2369), popular (0.1287), steep (0.068), rugged (0.0499), short (0.0415)

Query: The hiking trail is really [MASK].
Result: beautiful (0.0573), challenging (0.0512), steep (0.0474), good (0.0427), scenic (0.027)

Query: The hiking trail is extremely [MASK].
Result: scenic (0.2999), steep (0.0884), rugged (0.0806), popular (0.0655), difficult (0.0431)

Query: The restaurant is [MASK].
Result: closed (0.1758), vegetarian (0.0914), open (0.044), staffed (0.0372), haunted (0.0251)

Query: The restaurant is very [MASK].
Result: popular (0.1848), modern (0.0396), upscale (0.0261), small (0.0219), expensive (0.0197)

Query: The restaurant is really [MASK].
Result: nice (0.0929), amazing (0.0412), crowded (0.0376), good (0.0272), beautiful (0.0269)

Query: The restaurant is extremely [MASK].

Result: popular (0.255), expensive (0.0508), crowded (0.0258), modern (0.0226), successful (0.0224)

Query: The health clinic is [MASK].
Result nearby (0.1888), closed (0.1167), staffed (0.0987), open (0.0326), operational (0.0254)

Query: The health clinic is very [MASK].
Result modern (0.0602), small (0.0529), good (0.0439), busy (0.041), close (0.0306)

Query: The health clinic is really [MASK].
Result good (0.061), busy (0.0381), amazing (0.0327), nice (0.0273), thriving (0.0253)

Query: The health clinic is extremely [MASK].
Result busy (0.0828), poor (0.0793), small (0.0523), dilapidated (0.0507), crowded (0.0467)

Seed Question with Inputs:

includes the seed questions and an inputted response from a prior query.

Seed Question with Modifier Inputs: We present the results where the inputted response from a prior query is a modifier. Due to space limit, for each domain (apartment, hiking trail, restaurant, health clinic) we only present query results using one example modifier input.

Query: The [MASK] apartment was furnished.
Result entire (0.1981), whole (0.0706), penthouse (0.0639), upstairs (0.0353), downstairs (0.0171)

Query: The furnished apartment will be [MASK].
Result furnished (0.1017), renovated (0.0887), refurbished (0.0808), rented (0.0486), demolished (0.0432)

Query: The [MASK] hiking trail was paved.
Result entire (0.354), original (0.0372), main (0.0237), paved (0.0206), adjacent (0.0171)

Query: The paved hiking trail will be [MASK].
Result paved (0.0755), maintained (0.0455), removed (0.0442), restored (0.044), upgraded (0.0367)

Query: The [MASK] restaurant was nice.
Result whole (0.1677), entire (0.071), seafood (0.0173), italian (0.017), chinese (0.0131)

Query: The nice restaurant will be [MASK].
Result renovated (0.0897), reopened (0.0776), refurbished (0.0468), rebuilt (0.0383), demolished (0.0372)

Query: The [MASK] health clinic was open.
Result mental (0.0429), public (0.025), community (0.0214), local (0.0207), nearest (0.0181)

Query: The open health clinic will be [MASK].

Result renovated (0.0781), closed (0.0641), reopened (0.0531), opened (0.043), built (0.042)

Seed Question with Modifier and Action Inputs: We present the results where the inputted responses from a prior query are a modifier and an action, in order to elicit the actor. Due to space limit, for each domain (apartment, hiking trail, restaurant, health clinic) we only present query result using one example modifier and one example action input.

Query: The [MASK] renovated the furnished apartment.
Result owners (0.0681), couple (0.0539), owner (0.0423), family (0.022), landlord (0.0206)

Query: The [MASK] restored the paved hiking trail.
Result state (0.0478), volunteers (0.0412), department (0.0352), park (0.032), owners (0.0319)

Query: The [MASK] refurbished the nice restaurant.
Result owners (0.1122), owner (0.0547), company (0.0422), municipality (0.0349), club (0.029)

Query: The [MASK] restored the open health clinic.
Result government (0.1265), earthquake (0.028), city (0.0232), municipality (0.0186), legislature (0.017)

Notably, these highest-confidence responses to the queries using the Masked Language Model represent important, yet general starting points needed to distinguish the seed elements. For example, in the apartment-finding domain, important characteristics people pay attention to do include whether the apartment is furnished, modern, spacious, etc., or whether it contains a balcony. Others, such as whether the apartment contains a basement may be less relevant. Actions that are highly related to an apartment, like furnish, renovate, vacate, as well as actors that perform these actions, such as owners, landlords, etc, are also relevant, whereas other actors, such as couples, may be unnecessarily specific.

To summarize and answer the **RQ**, extraction using word embeddings with MLM finds associated actors, actions, modifiers and entities for constructing the domain model, by using seed question templates and reusing prior query results.

4.4 Discussions

We now discuss the significance of the results, their implications, as well as limitations and challenges associated with the approaches.

4.4.1 Comparison of Other Domain Knowledge Modeling Approaches

Both the previous approach to model domain knowledge using user-authored scenarios mentioned in Section 3, and this approach to model domain knowledge using MLM, offer different advantages and disadvantages. The extraction using typed dependencies and a scenario corpus has the advantage of discovering a diverse and highly relevant set of actions for each entity. However, that approach requires that one already have a corpus with scenarios to support the extraction.

Alternatively, the extraction using Masked Language Model finds associated actors, actions, and modifiers for constructing the domain model without a corpus by using the seed question templates and reusing prior query results. The disadvantage is that the results obtained using the MLM are limited to high-confidence words that may not provide the same kind of diversity and relevance to the seed noun phrase as what we see in the corpus-extracted results. It is likely that unless the corpus in the scenario extraction approach exists and has a large number of statements containing these domain knowledge, the approach may not be easy to find the domain knowledge required to build the domain models like we do using the second approach. We leave the detailed evaluation to future work.

Moreover, the two approaches can be complementary in building a domain model. The domain knowledge extracted from user-authored scenario with typed dependency can be used as inputs to the seed question templates in the second approach, in order to extract more complicated and relevant domain knowledge using Masked Language Model. For example, we found the noun "area" and its verb "search" with the first approach, and may use a seed question like "I want to search for an [MASK] area" in the second approach, to elicit modifiers that would describe the noun "area". Knowledge learned from using the Masked Language Model may also be used to give specific directions to authors when they author their scenarios. For example, we learned that "dilapidated" is a property describing the apartment, and may ask the author to write a scenario about his preference of the outlook or state of the apartment he is searching for.

Limitations in human knowledge can impact the completeness of domain models extracted from user-authored scenarios, and motivate the utility of MLM-based models to support identifying gaps in domain models. Research in psychology has revealed theories on *cognitive bias*, such as when individuals frame their next thought from their last thought, called *anchoring bias*, and *availability bias*, which states that people tend to heavily weight their judgements toward most recent information [179]. These biases can lead users to author scenarios in ways to overlook important or related concepts. The MLM-based models may serve as a helpful, complementary approach to identify gaps in models extracted from such scenarios. In Tables 3.9 through 3.12, one can observe the incompleteness of domain knowledge across multiple authors. In Table 3.9, out of 20 authors, only five authors mention "budget," while eight mention "appointments" to see the apartment In Table 3.11, out of 16 authors, six authors mention "water," and four authors mention the "weather." The choice of why some authors bring up these topics when answering the question, and others do not, may be explained by anchoring, e.g., cost of an apartment can serve as an anchor upon which the need to plan a budget follows, and availability, where in a recent or memorable hiking experience was affected by bad weather.

We believe that the extraction method using the MLM can be used as a "bootstrap" technique in real world settings, rather than a stand-alone tool supporting sentence completion. In our opinion, MLMs cannot substitute human expertise completely because unlike human experts, MLMs lack the ability to understand and interpret the specific problem context and to accurately generate requirement-specific information. However, this approach can be used to enhance an existing domain model to check for missing, high-probability entities or actions associated with domain elements described such models. For instance, we may use MLMs to identify missing domain model elements in a requirements artifact, such as missing actors (e.g., owners or landlords in

an apartment or building scenario), and then proceed to ask subject matter experts whether those missing elements require additional elaboration.

4.4.2 Seed Question Enhancement

In Section 4.2.1, we introduced a few seed question templates that can be used to elicit domain elements, such as modifiers attached to a domain-specific noun. It is worth noting that these example question templates can easily be altered to elicit more domain elements. For instance, apart from the examples shown to elicit modifiers and entities associated with the domain-specific noun, we could also vary the seed question to elicit other elements such as actors, actions, etc., by using templates "I want to [MASK] in the apartment.", "[MASK] are in the apartment.", and so on.

In Section 4.2.1, we introduce the possibility of reusing prior query outputs as an input to the seed question template to yield a new query to further elicit information on a related domain concept. For instance, we examined the example of using the modifier "spacious" as a seed question input to elicit the action "renovated", and querying with both inputs to elicit the actor "owners". It is noteworthy that this process can be repeated until a domain model is formed to a level of satisfaction, and that repetition of words across queries might strengthen their relevance in the model. A potential future work direction is to determine when and how that satisfaction level is reached.

4.4.3 The Effect of Using Intensifiers in Seed Questions

In Section 4.3, we presented results of using intensifiers in the seed question templates. In an eyeball test of the three templates, with no intensifier or the intensifier "very" and "extremely", respectively, we observe that the third sentence using the adjective "extremely" was most likely to surface value-oriented adjectives that others may or may not agree with, whereas the first sentence surfaced less value-oriented and more agreeable adjectives. This could mean that using intensifiers would yield modifiers that are closer to user preferences that can turn into soft goals or quality requirements, enhancing the domain model constructed from this approach.

4.4.4 Domain-specific Masked Language Models

A challenge for generating the domain model from word embeddings is whether the seed question templates can be reused across domains. For instance, given the seed question "The apartment is [MASKED]", we can change the domain reusing the same seed question with a different noun phrase, clinic, to yield the updated template: "The clinic is [MASKED]." Can large embeddings learnt by models, such as BERT, adapt to domain-specific problems? In Section 4.3, we show the results of reusing seed question templates across domains. These seed questions appear to adapt well to different domains, but there is space for improvement, and it is not guaranteed that they would adapt well to domains that are more obscure and thus not be well learned by the Masked Language Model. In such a situation, when we need to obtain word embeddings for text data that are domain-specific, such as in domains like legal, medicine, etc., we can still utilize the Masked Language Model based on BERT. One way is to treat this task as a downstream task to fine-tune BERT. By continuing to train the pre-trained BERT model with some of the domain-specific text

data, we can produce a word-embedding tuned to the specific domain [180]. Another way is to train the BERT-based model from scratch on a domain-specific corpus. This may result in better domain-specific word embeddings, but requires much more training data and computational power. Examples of using this approach include SciBEERT [23], which was trained for scientific text, and BioBERT [123], which was trained for medical text.

Moreover, when applied to less popular domains, the generation of the domain model from user-authored scenario approach may not be very effective, as users of these domains can be fewer, hence the authored scenarios obtained from fewer users may yield less domain knowledge for domain model generation. To address this limitation in the workplace, one could replace Amazon Mechanical Turk workers with employees to transfer the approach to an industrial setting, because employees may generally hold more domain-specific knowledge for systems they are developing. Although industrial settings may already possess domain models, these domain models can be bootstrapped by the generation from MLM approach, as this approach may enhance the model with knowledge that human-beings, such as employees, may tend to overlook, as discussed in Section 4.4.1.

4.5 Summary

In this chapter, we examined an approach to extract domain models from word embeddings using masked language models (MLM). The approach is based on prepared statement templates where two or more slots in the templates are filled by seed knowledge, such as a noun phrase, and a mask, which represents the query to which the model provides a response. The highest confidence responses in the MLM can be used to build out the domain model. The data and code are available online [206]. We compare this approach to the domain knowledge modeling using user-authored scenarios approach to understand how they differ and how they could complement each other. We leave the detailed evaluations to future work.

Chapter 5

Guided Preference Extraction in Interviews

In previous chapters, we have studied methods to model domain knowledge, in order to identify and obtain requirement-related domain elements. With these requirement-related domain elements we obtained, we now explore using them to refine requirement artifacts. In this chapter, we choose interviews as the requirement artifacts to refine by using requirement-related domain elements we extracted from word embeddings using an MLM, as described in Chapter 4. While our automated domain elements and preference extraction from MLM and scenarios in previous chapters have demonstrated promising capabilities, interviews remain essential in the requirements engineering process for several critical reasons, including but not limited to a chance to validate and disambiguate extracted requirements, capture tacit knowledge that may not have been extracted previously, and to prioritize requirements. We believe that automated extraction and elicitation techniques remain a complementary approach to stakeholder requirements gathering, as opposed to acting as substitutes.

5.1 Motivation and Goal

Interview techniques for uncovering stakeholder needs are important, because interviews serve as an interactive way for requirement analysts to directly engage with stakeholders [46] to elicit their preferences. By asking targeted questions and gathering responses during interviews, interviewers are able to understand the problem domain and delve below surface level needs to learn the individualized likes, dislikes, priorities, and tradeoffs of interviewees [34]. In addition, the interviewer can resolve ambiguity in a stakeholder's statements and uncover additional detail and tacit knowledge through follow-up questions [31, 63]. Moreover, we focus on eliciting stakeholder preferences, which are qualities or characteristics of a software system that a stakeholder desires. Stakeholder preferences are often tacit knowledge, yet maximizing these preferences play an essential role in successfully building a software system tailored to suit stakeholder needs, and proper interviews are prone to uncovering tacit knowledge.

Despite the benefits of interviews, they are not easy to conduct. Skills of individual interviewers can vary significantly, such as interpersonal skills needed to create rapport with interviewees, cultural and language barriers may exist between the interviewer and the interviewees, and a lack

of shared understanding in the interview domain can exist, further causing the quality of elicited needs to vary [39, 62, 202]. Psychological difficulties can also negatively affect interview elicitation quality. Difficulty coming up with appropriate follow-up questions in real time [17], tunneling or getting lost in a single line of inquiry [53], asking an excessive number of vague, irrelevant or off-topic questions and forgetting to come back to previously raised points [31], jumping between topics or failing to probe topics when necessary [29], and anxiety or self-doubt, are a few examples. Hence employing a systematic approach to interviews may be beneficial.

Since interviews naturally produce a large amount of natural language data, advances in natural language processing (NLP) techniques can be helpful in guiding, improving and partially automating the original manual interview elicitation and analysis process [55, 70, 153, 158, 165]. Neural models in particular have shown strong performance across many NLP tasks. Constructing these models, however, can require millions to billions of data points. Transfer learning, wherein information is learned in one or more tasks and later transferred to another domain or task, has reduced the need for large datasets. This is particularly true of word embeddings, which are often constructed using unsupervised machine learning methods over large corpora. Currently, interviews exchange a wealth of information, and requirements engineers have historically used note-taking or their memory during the interview to record information for recall during the interview [158]. We believe improvements to interviewing can be obtained by minimizing the engineer's reliance on incomplete notes and memories by using recent advances in NLP.

Domain models, which describe knowledge about specific application domains [26], are built to capture requirements for systems. Techniques for automatically extracting domain model elements can include using rules based on information retrieval, using natural language dependency parsing 3, and so on. Supervised machine learning methods can be used to recommend relationships between entities in a domain model [20]. Previously, we have discovered that domain models can be extracted via word embeddings using masked language models (MLM) in Section 4. This suggests a potential to use MLM to suggest related topics to interviewers based on the interviewees' responses during real-time interviews, which may in turn alleviate some of the challenges mentioned. Specifically, the suggestions made by the MLM may partially enhance the interviewer's domain knowledge and help them come up with appropriate follow-up questions more quickly and easily. This in turn enable interviewers to guide interviewees in probing deeper into a topic of interest during an interview. In this chapter, we report on our work to embed MLM into a real-time guided interview elicitation method, to further automate the interview elicitation process. We also add real-time interview transcription functionality into the method, and use part-of-speech tagging to highlight potential concepts of interest in the transcription, to help interviewers more easily visualize potential topics of interest and generate suggestions with MLM, and to potentially reduce the possibility of the interviewer going off-topic for a long time and forgetting other previously mentioned topics. The MLM suggestions along with the transcription may also ease the cognitive load of the interviewer and help them manage the overwhelmingly rich information in the interviews. Our methods include: 1) we build a back-end MLM-based model that generates domain elements for a selected keyword during interviews, 2) we build a guided interview elicitation method with real-time interview transcription, related topic suggestion, and potential concept highlighting functionalities using our back-end model, 3) we design and conduct a controlled experiment consisting

of 32 interviews with stakeholders across four directory service domains to evaluate the effect of the method on improving elicitation.

5.2 Research Methodologies

We describe our research methodologies in this section below.

5.2.1 Domain Elements Generation

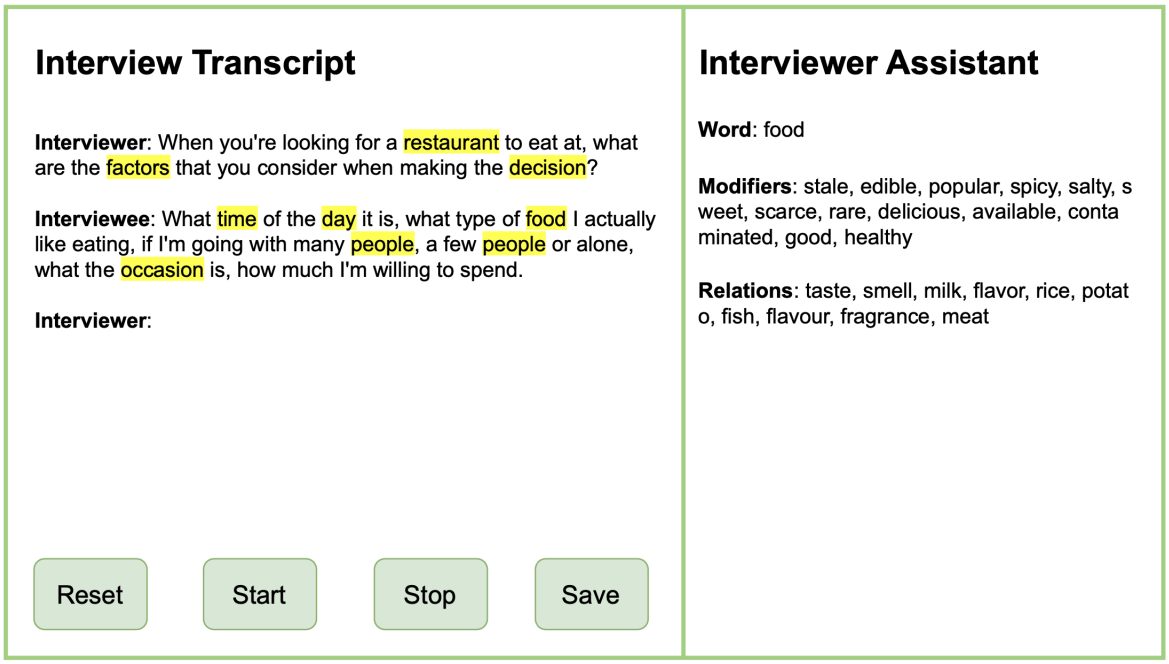

Figure 5.1: Tool Mockup

The interviewer's domain knowledge and subject matter expertise are vital in the success of interview elicitation. Moreover, it would likely influence the interviewer's ability to come up with the appropriate follow-up questions in real time, which may, in turn, cause failure to probe into topics of interest appropriately. Along with the interviewer's overwhelming cognitive load and information overload, the resulting interview data may lack conceptual depth and breath.

In Chapter 4 we have studied the efficacy of extracting domain models from word embeddings using masked language models (MLM). The approach begins with an initial entity, called the *seed entity*, and one or more typed questions used to generate additional entities in relation to the seed entity. The question type corresponds to the relation type that links the entities in the model. The questions are expressed using a prepared template that includes the entity expressed as a domain-specific noun (e.g., apartment, restaurant), a verb (e.g., has, is, be) and the mask [MASKED] corresponding to the related entity, modifier, or action to be extracted. For example, the template

"The entity has a [MASK]." is used to extract entities that are in a part-whole or meronymy relation to the seed entity. This yields entities such as kitchen, balcony or gymnasium for the seed entity "apartment", which could mean a single apartment or apartment building. The MLM results are ranked from highest to lowest confidence score computed from the model. The confidence score is the log-likelihood associated with the predicted filler word for the mask. They also found that using intensifiers, such as "very" or "extremely", in the question templates, can elicit more value-laden results.

We adapt this approach to build a back-end model that automatically generates domain elements related to a seed entity input, which is a preference-related keyword selected by the interviewer based on the interviewee's speech during an interview. Details about the selection process are explained in Section 5.2.1. The model then suggests the related domain elements to the interviewer in real-time during an interview. The interviewer can then decide whether to follow-up on the generated element while discussing the same topic. We hypothesize that integrating related domain elements into the discussion in real-time reduces the likelihood of the interviewer being overwhelmed by cognitive load as topics change over time. We choose the following question templates to elicit modifiers and associations related to the selected entity in the interview.

Question to elicit modifiers

We use the question templates below to elicit modifiers attached to the selected domain-specific seed entity, denoted as {entity} using curly brackets. Eliciting modifiers can lead interviewees to identify additional entities that relate to the seed entity.

Query: I want a(n) {entity} that is [MASK].
Query: The {entity} is extremely [MASK].
Query: When does the {entity} become [MASK]?

Question to elicit relations

We use the question templates to elicit part-whole and other meronymy relations attached to the chosen domain-specific noun phrase, denoted as {entity} in curly brackets.

Query: The {entity} has a [MASK].
Query: The [MASK] is the most important part of the {entity}.

The MLM we use is a model from HuggingFace, pretrained on a union of five large English corpora [119]. The MLM is asked to fill in the [MASK] positions with five most likely substitutes based on the model's confidence score. For example, when the {entity} is replaced with "kitchen", running through the question templates above would result in generating the following domain elements:

Modifiers related to kitchen are:
'modern', 'spacious', 'unbearable', 'clean', 'perfect', 'dark', 'large', 'comfortable', 'colder', 'heated', 'amazing', 'beautiful', 'boring', 'small'

Relations related to kitchen are:
'pantry', 'kitchen', 'stove', 'bathroom', 'dairy', 'fireplace', 'refrigerator', 'garden', 'cellar'

5.2.2 Real-time Guided Preference Elicitation

We realize real-time machine-guided elicitation in a tool that consists of the following components: a real-time audio transcription service, an MLM to generate doamin elements, a separate log-in page for the interviewer and the interviewee, and video conferencing support for the interviewer and interviewee to see each other during a remote interview. We discuss these components, and how they are connected, below.

Audio Transcription Service

We use a real-time audio transcription service that renders the interviewer's and interviewee's speech into text during the interview, while distinguishing the source of the audio input in order to separate the interviewer's transcription from the interviewee's transcription. The separation allows us to apply any number of transcript analysis techniques to either of the two transcripts independently. For example, using the interviewee's transcript, we can identify noun phrases spoken by the interviewee to be analyzed by MLM. A mockup of the transcription service is shown in Figure 5.1, on the left side.

Domain Element Suggestion

As discussed in Section 5.2.1, we build a back-end model that can automatically generate and suggest domain elements related to a preference-related keyword. In the interviewer interface, the interviewer is able to choose any preference-related keyword mentioned by the interviewee, by clicking on the keyword in the audio transcription display. The interviewers can use their judgments to decide which interviewee-raised concepts are requirements related and are suitable for follow-up questions during the interview. A mockup of the suggestion can be seen in Figure 5.1, on the right side.

To better help the interviewer pinpoint the preference-related keyword to click on during the interview, and to reduce distraction likely caused by the large amount of audio transcription text, we use a part-of-speech tagger to tag and highlight the nouns in the text transcriptions, so they are easier to spot, as can be seen in the mockup, highlighted in yellow.

Seperate Log-in Pages

We provide separate log-in pages for the interviewer and the interviewee. The interviewer, once logged in, is able to access full-functionality of the interview tool, with audio transcription service and the domain element suggestion service. On the other hand, to minimize distractions for the interviewees and allow them to focus on the interview itself, the interviewee, once logged

in, will not be able to see the displays of audio transcription or domain element suggestions. The interviewer and the interviewee will use different credentials to log into our tool.

Video Conferencing

We provide compatibility with video conferencing for people wishing to conduct interviews remotely. Both the interviewer and the interviewee may use Zoom or Microsoft Teams, for example, to turn on the camera and share video without interfering with the operation of the elicitation method.

5.2.3 Evaluation Methods

In this study, we are interested in whether the real-time machine-guided elicitation (RTMGE) method can increase requirements coverage beyond a baseline method, which includes evaluating whether interviewees who participate in interviews with the RTMGE method mention more preferences or whether they are more specific and less abstract about the mentioned preferences. We acknowledge that collecting requirements with increased coverage and specificity are not all aspects to successful interviews, yet we believe these two aspects are important in ensuring that the interviews are advancing requirements elicitation in a meaningful way. We evaluate the RTMGE by testing the following hypotheses (the null hypotheses, not shown, assume no effect).

H1 Interviews conducted with the RTMGE yield a larger number of unique concept mentions by interviewees.

H2 Interviews conducted using the RTMGE yield more specific concept mentions by the interviewees.

The null hypotheses for H1 and H2 are that there is no significant difference between interviews conducted with and without RTMGE, and a p-value is calculated with their respective suitable statistical tests to reject the null hypothesis with the chosen significance level $\alpha = 0.05$. We talk about our experiment design and statistical tests below.

To test these hypotheses, we conducted a controlled experiment. The experimental design was reviewed by our University's Institutional Review Board (IRB) to minimize risks to human subjects who must consent to participation prior to joining the study. We now discuss the process for hiring and training interviewers, recruiting interviewees, and analyzing and reporting the data.

Hiring and Training interviewers

In this study, we hired eight interviewers, who were current students in the Master of Software Engineering (MSE) Program at Carnegie Mellon University. Prior to acceptance, MSE students must have at least two years of software engineering experience in full-time software engineering positions, which typically includes frontend and backend development. All interviewers had completed two courses on requirements engineering and product management, covering topics in identifying customer value and in modeling and analyzing requirements. In addition, each interviewer was trained by the investigators requirements elicitation using Ferrari et al.'s course,

entitled "Learning Requirements Elicitation Interviews with Role-playing, Self-assessment and Peer-review" [61].

The elicitation training consists of the following steps: 1) *preliminary training*, wherein the interviewer watches the training video on YouTube created by Spoletini, 2) *first role-playing interview*, the interviewer is given a product description and asked to conduct a 1-to-1 interview about a product in a role-playing environment as requirements analyst; 3) *mistake-based training*, wherein the interviewer watches a YouTube video on common mistakes and recommendations; 4) *self-assessment*, wherein the interviewer is required to listen to their own interview recording from the prior interview, and to complete a self-assessment questionnaire about the mistake types explained in the mistake-based training stage; 5) *second role-playing interview*, wherein the interviewer is given a different product description and asked to conduct an interview using the same approach described in the first role-playing interview; and 6) *self-reflection*, wherein the interviewer self-assess their second interview using the same approach described in the first self-assessment, after which they complete a self-reflection questionnaire about their experience. Following the recommendation of Ferrari et al. on how to shorten the course and focus on role-playing interviews, which according to their presented study are the most useful and effective activities, we removed the peer-assessment step, but retained the two self-assessment steps. For role-playing, the interviewer trained with us where we played the role of instructors. The training time was approximately 3.5 hours for each interviewer.

Finally and in accordance with the IRB, all interviewers were required to complete training on the protection of human subjects, since the interviews would be used for research purposes.

Recruiting Interviewees

Each interviewer was assigned to four 20-minute time slots to conduct interviews, which yields a total of 8 x 4 = 32 total time slots. Interviewees were recruited using e-mail and invited to sign-up for one of the 32 time slots. Each interviewee must be at least 18 years old, have experience using a web or mobile application, and be proficient in English. Each interviewee also signed a human subjects research consent form regarding their rights and personal data protection. Participating interviewees were paid a $25 Amazon Gift Card as compensation.

Interview Design

Following the work in Section 4 and to study the RMGTE across different domains, each interview was assigned to one of four directory service domains. The interviewers were provided with a general question upon which to base their interviews corresponding to the assigned domain as follows:

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?

The eight interviewers were each assigned to either a control (C) group or a treatment (T) group for the duration of all four interviews to avoid cross-contaminating results due to knowledge

transfer arising from using the RTMGE method. The four treatment group interviewers were able to access the full functionality of the tool, including audio-to-text transcription, highlighting of candidate preference words, and suggestions for related domain concepts. The four control group interviewers used a simplified version of the tool, wherein only the audio-to-text transcription is shown without highlighting and without domain concept suggestions. To further avoid cross-contamination, the interviewers were told not to communicate with each other about the interview study, and they did not know which group they are assigned to. Each interviewer was given an individual instruction about how to conduct their interviews tailored to their group. The treatment group interviewers were provided a chance to become familiar with the highlighting and domain concept suggestion features of the tool prior to their first interview. None of the interviewers or interviewees know about the purpose of the study, or the research questions.

Each interviewer conducted one interview in each of the four domains without repetition to reduce learning effects from conducting multiple interviews in the same domain. The order of interview domain assignments to each interviewer was further randomized to minimize learning affects due to preceding or following one domain with another (see Table 5.1). The first column corresponds to the interviewer ID and whether the interviewer was in the treatment group or in the control group. The following cells indicate the different interview domains in the order of assignment. The second to the last column correspond to the rounds, where round i is the i-th interview conducted by each interviewer. For example, interviewer T1, which corresponds to interviewer 1 from the treatment group, interviewed on the domain *apartment* in the first round; interviewer C3, which corresponds to interviewer 3 from the control group, interviewed on the domain *hiking* in the fourth round; and so on. The randomized assignment process has the following constraints: 1) each interviewer's four rounds (the rows in the table) should have four different domains; and 2) within each round (the columns in the table), the interviewers of the same group should interview on four different domains, i.e., each round spans four different domains for all treatment group interviews and four different domains for all control group interviews. By randomizing the order of the interview domains across interviewers, we are distributing any learning effects, wherein the interviewer may become more familiar with interviewing as they interview more people, causing the domains covered in later interviews to produce better outcomes than the domains covered in earlier interviews.

All interviews were conducted remotely in this study. Participants used Zoom to turn on their cameras to video chat with each other during interviews.

After each interview, the transcriptions were recorded and de-identified prior to analysis.

Transcript Analysis

We test hypothesis H1 by analyzing the transcripts to identify the number of domain concepts mentioned in each interview by the interviewees. In the first step, all transcripts were randomly shuffled to remove any indication of which group (treatment or control) the transcript belongs to. Second, we read each transcript in the shuffled order, and recorded unique preference-related concepts representing stakeholder needs mentioned by the interviewee in each turn in the interview, wherein a turn is defined as when the interviewer and the interviewee each finish speaking. If a

Table 5.1: Interview Schedule

Interviewer	Round1	Round2	Round3	Round4
T1	1	4	2	3
T2	2	1	3	4
T3	4	3	1	2
T4	3	2	4	1
C1	3	1	2	4
C2	2	3	4	1
C3	4	2	1	3
C4	1	4	3	2

The topic IDs are assigned as apartment=1, restaurant=2, hiking=3, clinic=4.

concept was mentioned more than once, it was only counted once per turn. Another person independently verified the results and reached an agreement with the previously identified concepts. Next, we tallied the unique concepts mentioned per interview using the per-turn tallies. A concept is determined as not unique if it is mentioned more than once. The per-interview tallies provide a measure of unique concept mentions independent of interview length. In both stages of identifying unique concepts (per-turn, and per-interview), we only tally and retain the unique concept phrases that were the most discriminative and specific. For example, if in the apartment domain the concepts "utilities", "water", "electricity" are mentioned in the same turn or interview, we only kept the latter two concepts, since both are kinds of utility and thus are more specific.

After identifying the unique concepts, the data was then used to observe the unique preference-related concepts elicited by each group. Within each group and each domain, we calculated 1) *per-turn average*: the number of unique preference-related concepts elicited from the interviewee each turn, divided by the number of turns in the interview. This was done as follows: for each turn, we counted the number of unique preference-related concepts elicited from the interviewee, denoted as c_i, averaged by the total number of turns in the interview, denoted as n. That is, for an interview, we calculated $\frac{\sum_{i=1}^{n} c_i}{n}$, which represents the average number of unique preference-related concepts elicited per turn in the interview. We did this for all interviews in each of our four domains, averaged by the number of interviews in the domain, and compared the results for the control and treatment group. 2) *per-interview average*: the number of unique preference-related concepts elicited from the interviewee in each interview, divided by the number of interviews for each domain topic. Note that this is different from the per-turn average, because interviewees may speak about the same concept across several turns. This analysis provides an understanding of interview-level elicitation outcome, of whether the treatment group elicits more or less unique preference-related concepts in each interview, compared to the control group.

After we calculated the per-turn and per-interview averages, we performed 3) *statistical significance tests*: We first checked the distribution of the data for normality using the Shapiro-Wilk normality test. Because the data distribution is not normal, we used a Wilcoxon ranked sum test, also known as the Mann-Whitney U test, on the per-turn average and per-interview average to test

hypothesis H1 by testing whether there is a statistically significant difference between the mean number of unique concepts mentioned between the groups. We opted for the Wilcoxon ranked sum test since it is a non-parametric test that compares the ranks of observations between two groups, and does not assume normality. To ensure interviewer style does not affect the statistical tests independence assumption, we group the results by the same interviewer together. We arranged our data for the tests in the format that the first column contains the resulting average turn-based counts or the interview-based counts in a given interview, the second column contains the group id denoting control group as 0 and treatment group as 1. We adopt the following null hypothesis:

Hypothesis H_0 (null hypothesis): There is no statistically significant difference between the groups.

We choose $\alpha = 0.05$ as the significance level. If our test results in a p-value smaller than the chosen significance level, then the observed data is unlikely to have occurred if the null hypothesis were true, hence nullifying the null hypothesis, and the result is considered as statistical significant. On the other hand, a p-value greater than the chosen significance level does not indicate statistical significance.

We show our results for each of these analysis methods in Section 5.3.2.

Ontology Construction

We test hypothesis H2 by measuring the specificity of unique concepts mentioned by the interviewees in each interview using a constructed ontology. After the transcript analysis, the unique concepts from each interview within a single domain were pooled into a single list without repeats. Next, the list of concepts was randomly shuffled, and then the analyst compared concepts to identify relations between concept pairs. Based on the MLM questions introduced in Section **??** that probe the model for hypernymy and meronymy relations, we applied each of the following tests to concept pairs: 1) is one concept a "kind of" the other concept, or is there a hypernymy relation between the two concepts; 2) is one concept a "part of" another concept, or is there a meronymy relation; and; 2) if the two concepts are similar, what is the concept that generally describes the similarity and is it in the list of concepts, or is the third concept a hypernym of the two concepts.

We represent the ontology logically as follows: for a concept pair c_1, c_2, there may exist a transitive, asymmetric hypernymy relationship $kind_of(c_1, c_2)$ if and only if c_2 is a hypernym of c_1, or there may exist a non-transitive, asymmetric meronymy relationship $part_of(c_1, c_2)$ if and only if c_1 is a meronym of c_2. The concepts and relations can be represented as an acyclic, directed graph (a tree) in which each concept represents a unique node and each relations between two concepts represents an edge between two nodes. In the tree, the root concept is the domain word, and the leaf nodes are the most specific concepts in the ontology. For example, in the restaurant domain, the root is "restaurant" and "environment" may be a leaf node to "restaurant", and a parent node to "music," "noise level," and "lights." Below is an abbreviated illustration of a constructed concept ontology tree, in Figure 5.2.3.

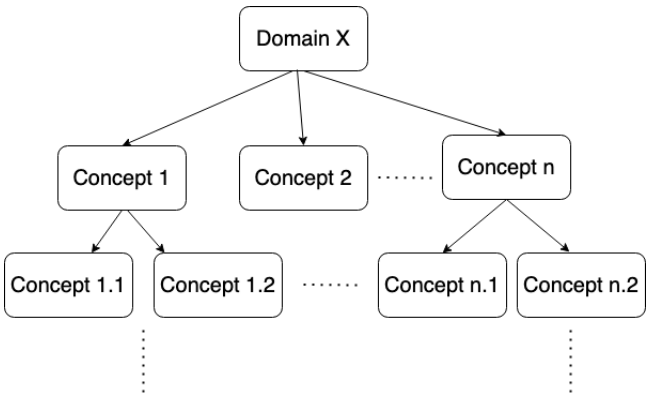

Figure 5.2: Illustration of the Ontology Tree

We constructed a separate ontology for each of the four domains. For each concept that appears in a transcript, we calculated the tree depth of the concept node in the ontology, and observed whether there is a difference in depth for concepts elicited from the control versus treatment group, since a concept at a larger tree depth represents a more specific concept than one from a smaller depth. To analyze the ontological differences between groups, we next computed 1) the *weighted average tree depth* by counting the number of unique concepts c_d per group at depth $d = 1, 2, 3, ..., h$ for the height h of the tree, and dividing by the number n of interviews in the group, which is equal to $\frac{\sum_{d=1}^{h} c_d \times d}{n}$; and 2) *statistical significance tests* using Student's t-test, since this data passed the Shapiro-Wilk test for normality, to see if the difference in tree depths between groups within a domain is statistically significance. We chose the weighted average tree depth for this test because it provides greater weight to concepts with more specificity on a linear scale. This means that concepts at a larger depth, which are more specific, shall have a larger weight in our statistical analysis. Again to ensure interviewer style does not affect the statistical tests independence assumption, we group the results by the same interviewer together. We present our result in Section 5.3.3.

5.3 Results

We present our elicitation, interview transcript analysis and ontology construction results for our research study, mentioned in Section 5.2.3, below.

5.3.1 Elicitation Outcome

Our research study has elicited in total 29 transcripts out of our 32 interviews conducted, 15 from the control group and 14 from the treatment group. We discarded one of our interviews from the control group due to the interviewer forgetting to save the transcript when interviewing about the apartments domain. We discarded two of our interviews from the treatment group, one due to the recruited interviewee having no knowledge of the domain (i.e., hiking) in which he is being interviewed on, and thus they could not answer domain-related questions at all, and the other due

to an interviewer unable to complete their last interview. Because these discards occurred after the interviews were conducted, our effort to minimize ordering effects in the scheduled order of the interviews assigned to each interviewer remained unaffected.

5.3.2 Interview Transcripts Analysis

We conducted the quantitative analysis methods mentioned in Section 5.2.3. Below we show our results to each of the methods.

Per-turn Interview Result

Recall from Section 5.2.3, a turn ends when both speakers finish speaking once. For example, this may arise when the interviewer finishes asking a question and when the interviewee finishes answering. We present the per-turn interview elicitation result for each of our four domains below in Table 5.2, with the higher number highlighted.

Group	Apartment	Restaurant	Hiking Trail	Clinic
Control (C)	1.6844	1.5749	2.101	1.4123
Test (T)	**1.9942**	**2.3762**	**2.8337**	**1.7810**

Table 5.2: Turn-Based Elicitation Result By Group

As can be observed in the table, the treatment group has a higher average number of unique preference-related concepts elicited per turn than the control group, in all four domains. When all four domains are combined, the treatment group shows an average **0.5532** more unique preference-related concepts extracted per turn than the control group. We ran the Wilcoxon ranked sum test (large effect size) to test statistical significance of our result, and obtained a p-value of 0.056 slightly greater than 0.050, which indicates that we fail to reject the null hypothesis. This suggests that there is insufficient statistical evidence to support our alternative hypothesis, though the result approaches significance. Therefore the RTMGE method does not cause a statistically significant improvement for the per-turn interview elicitation outcome.

Per-interview Average Result

We present the per-interview elicitation result, mentioned in Section 5.The result is shown in Table 5.3, with the higher number highlighted.

Group	Apartment	Restaurant	Hiking Trail	Clinic
Control (C)	31.0	30.0	31.25	**18.75**
Test (T)	**35.0**	**37.6667**	**38.6667**	18.0

Table 5.3: Interview-Based Elicitation Result By Group

Domain Topic	total # nodes	Depth 1 Concepts
Apartment	139	affordability (11), amenity (24), atmosphere (4), building style (8), building system (3), commute (3), lease (10), management (18), neighborhood (6), review (5), safety (4), technology (5), unit (19), utility (4)
Restaurant	140	affordability (12), availability (4), dining option (6), environment (10), food (23), location (8), menu (7), occasion (12), online information (10), personal condition (10), restaurant style (15), service (10)
Clinic	98	affordability (6), availability (10), environment (12), location (5), personal condition (3), recommendation (8), setup (3), staff (18), technology (8), treatment (14)
Hiking Trail	138	affordability (4), amenity (9), availability (2), environment (10), location (7), online information (12), personal condition (12), provision (21), safety (11), setup (2), trail detail (11), view (17), weather (6)

Figure 5.3: Number of Nodes and Depth 1 Concepts for Each Domain

		# Interviews	d_1	d_2	d_3	d_4	Total
Apartment	Control (C)	3	14	50	12	0	76
	Test (T)	4	14	65	35	1	115
Restaurant	Control (C)	4	11	51	25	0	87
	Test (T)	3	12	60	24	4	100
Clinic	Control (C)	4	9	30	21	0	60
	Test (T)	4	10	32	27	3	72
Hiking Trail	Control (C)	4	13	58	19	0	90
	Test (T)	3	12	57	30	2	101

Table 5.4: Ontology Tree Depth by Group

As can be observed in the table, the treatment group elicited more unique preference-related concepts per interview for three of the four domains, with the exception of the clinic domain where the result of the control group is slightly higher. On average, the treatment group elicited 4.59 more unique concepts than the control group. We ran the Wilcoxon ranked sum test (medium effect size) to test statistical significance of our result, and obtained a p-value of 0.156 greater than 0.050, which indicates that we fail to reject the null hypothesis, and therefore the RTMGE method does not cause a statistically significant improvement for the per-interview elicitation outcome.

5.3.3 Preference Concept Ontology Construction

We constructed four preference concept ontologies mentioned in Section 5.2.3, one for each domain that we studied. Each node in an ontology represents a unique preference concept, with the top node being the root domain concept (apartment, restaurant, clinic, hiking trail), and nodes with larger tree depths represent more specific concepts than their parent nodes. An example of a part of the restaurant domain ontology tree is displayed below in Figure 5.3.3. In this example, "restaurant" is the top node, "food" is one of the depth=1 nodes, "food taste" are among the depth=2 nodes under "food", and "spiciness" is a depth=3 node under "food taste".

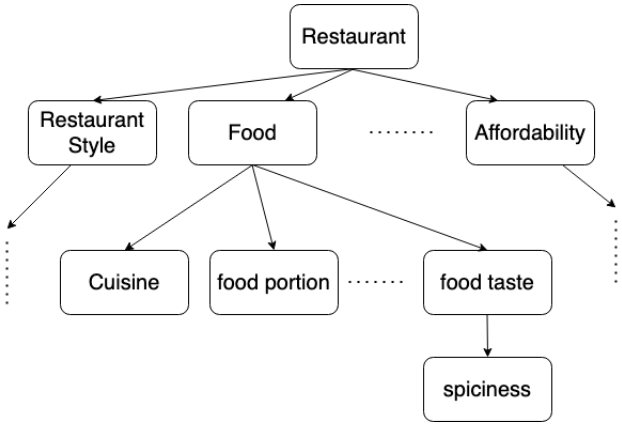

Figure 5.4: Sample Part of Restaurant Ontology Tree

A list of the subsumption relationships $sub(C_1, C_2)$ we discovered in our ontology construction is below:

1. "A is a type of B": Concept A belongs to a broader category represented by Concept B. For example, "waterfall" is a type of "unique scenery" on a hiking trail.
2. "A is an aspect of B": Concept A represents a specific characteristic or component of the broader concept represented by Concept B. For example, "decoration taste" is an aspect of "building style" when looking for an apartment.
3. "A is a property of B": Concept A describes some feature or property of Concept B. For example, "pet friendliness" is a property of "amenities" within an apartment.
4. "A is an option of B": Concept A represents a potential choice or alternative that is available within the context of B. For example, "monthly basis lease" is an option of "lease length" when signing a lease with an apartment.
5. "A is a part of B": Concept A is a component or constituent that contributes to the structure, function, or composition of Concept B. For example, "treatment area" is a part of the "environment" of a clinic.

Furthermore, in the example above in Figure 5.3.3, "cuisine", "food portion" and "food taste" are parts of "food", while "spiciness" is an aspect of "food taste".

Due to the large size of the ontologies,[1] we only present statistics including the total number of nodes for each domain ontology, as well as their respective depth=1 concepts that represent the most general concepts elicited in the domain (see Figure 5.3.2). In the parentheses after each depth 1 node concept, we show the number of nodes under that concept.

Below in Table 5.4 we show for each domain the number of interviews and the number of nodes at each tree depth d_i for each group. As can be observed from Table 5.4, the treatment group mostly exhibits a larger number of concepts elicited at each depth than the control group, even in cases where the number of interviews in the treatment group were fewer than that in the control group. In addition, the treatment group exclusively resulted in depth=4 concepts.

The Student's t-test (large effect size) shows a p-value of 0.0653 greater than 0.050, indicating that our result is not statistically significant.

5.3.4 Answer to Hypotheses

In response to our **H1**, which tests whether interviews conducted with RTMGE yield a larger number of unique concept mentions by the interviewees, we first conducted a turn-based average elicitation analysis, an interview-based average elicitation analysis, and statistical significance tests to test our analysis results, as shown in Section 5.3.2. We have observed that our method was able to increase the number of unique preference-related concepts elicited both per turn and per interview, for all four chosen domains, except that the number of unique preference-related concepts elicited per interview for the Clinic domain shows a slightly higher result in the control group. However, our results did not reach statistical significance; hence we cannot reject the null hypothesis because the observed differences could be due to random variation rather than to a true effect of the RTMGE method.

We also constructed an ontology tree for each of the 4 domains in Section 5.3.3. To answer our **H2**, of whether interviews conducted using RTMGE produce more specific concept mentions by the interviewees, we have observed from the ontology construction result that the treatment group using our method is better at eliciting more fine-grained concepts than the control group, in terms of the total number of unique concepts elicited from all interviews from the group combined together. For example, in the restaurant domain, when mentioning the menu as an item of interest, the treatment group interviewees expanded into detailed topics such as item description, nutritional profile, pricing information, and visual display of menu items, going into even more detail about pricing by talking about pricing transparency, and whether tax is included in the price on menu, while the control group focused sorely on the menu item descriptions. Our result, however, did not reach enough statistical significance, hence we fail to reject the null hypotheses, again because the observed differences could be due to random variation rather than a true effect of the RTMGE method.

Overall, although the results trend in the right direction, statistical significance was not reached, and we could not conclude from our tests that the RTMGE method is effective in eliciting more unique requirements-related concepts, or concepts that are more specific.

[1]For the complete ontologies, see [207]

5.4 Discussions

We now discuss the results, including how well the method addressed the interview challenges, applicability to practice, and limitations and future work.

5.4.1 The RTMGE's Effects on Tackling Interview Challenges

Our results have not reached enough statistical significance to demonstrate their usefulness. However, since the statistics are underpowered due to small sample size, there is a possibility that with more samples collected, the results would approach or reach significance. We did not do this in this study because of the time-consuming and costly process of recruiting and training qualified interviewers, but we are hopeful that future research in this area can collect more samples, while making modifications to the existing pipeline. We discuss more about the limitations of the method in Section 5.4.3.

We now discuss how RTMGE may be able to address certain aspects of interview challenges that are previously mentioned. Although we did not formally evaluate this in the scope of our analysis, we hope to use this discussion to shed light on future research on how interview challenges can be tackled through elicitation guidance methods. We also discuss some interesting findings during our transcript collection process.

The RTMGE method allows interviewers to select entities in a transcript before suggesting concepts related to the domain during the interview. The suggestions serve to trigger or expand the interviewer's awareness of concepts in the domain while the interviewer is managing the dynamics of the interview. When an interview is demanding, the method has may cognitive load by reminding the interviewer of additional directions to explore for a given topic of interest.

Stakeholder preferences refine the main idea (e.g., a playground near an apartment), and they are highly subjective and contextually dependent on an individual's experiences. As such, the preferences of an interviewer and stakeholder may differ, while each may take these preferences for granted. The subjectivity of preferences can lead preferences to exist as tacit knowledge that is not recognized during an interview. The RTMGE's suggestion feature draws out this tacit knowledge to be explicitly discussed in an interview, which may allow interviewers to properly probe deeper into details for which the interviewer may be unfamiliar, which we call the *topic depth*. While suggestions may not be relevant, the interviewer can decide whether to prioritize the suggestion or they may be inspired to pursue other topics, further expanding *topic breadth*. The suggestion feature may nudge the interviewer and interviewee to be more exploratory, placing an emphasis on open-ended questions that the interviewer would not have otherwise thought to ask without using the method.

For example, during an interview about finding hiking trails, the interviewee mentions the word "incident" as part of their safety concerns. The RTMGE method suggests specific concepts such as "looting" and "kidnapping" that may have facilitated the interviewer and interviewee to discuss different incident types, including concerns for robbery, bear attacks, snake bites and chance of getting lost. In this regard, even if the interviewer does not use the specific suggestions, they may use the refined context as motivation to explore an otherwise vague concept. This discussion

covered the interviewee's wish for rangers present in certain hiking areas and new features, such as methods to receive hiker safety alerts. These preferences were not observed in the control group, wherein the interviewee only touched the subject of "incident" without further discussion.

In the restaurant finding interviews, where both the control and treatment groups mention menu item descriptions as qualities they care about, only the treatment group further discussed other aspects of the menu, such as nutritional profiles and pricing transparency. The treatment group interviewees exclusively mentioned different dining options, such as in-person dining, takeout, and delivery, and extensively discussed dimensions of delivery options, including delivery speed, order placing methods, as well as delivery fee, driver tips and platform charged service fee as part of their concerns for pricing transparency.

The features discovered through the RTMGE method can be used to develop more personalized software. In the hiking trail example, above, software could track evidence of hiker safety incidents or pull such evidence from public police reporting; ranger presence could be realized as a selection filter for users; and hiker alerts could be embedded in the app. In the restaurant example, users of a restaurant finding app could report their individual experiences on delivery speed, delivery fees and tipping, among others to increase price transparency for users who alter search for restaurants.

In the RTMGE's real-time audio transcription service, we use a part-of-speech (POS) tagger to tag and highlight nouns in the transcriptions, which allows interviewers to easily spot keywords as preference concepts around which they can choose to develop their discussions. This may ease the cognitive load that overwhelming information during an interview may have placed on the interviewer and reminds the interviewer of topics that have previously been mentioned or discussed. Through this functionality, the interviewer may be able to keep track of what topics are remaining to be discussed, which may help interviewers better organize their interviews, and ease the potential challenge of interviewers spending too much time on one topic (i.e., tunneling) or too much time off-topic, or forgetting to return to previously mentioned topics of interest.

5.4.2 Applicability to Requirements Practice

The RTMGE method has been developed, tested, and deployed with the Zoom meeting platform. The implementation's architecture consists of three components: (1) a process for recording and transcribing audio; (2) a process for analyzing transcripts to report topics of interest for discussion; and (3) a dashboard that allows the interviewer to start/stop the processes and to read the transcript and topic analysis in real time. Our deployment of the RTMGE method involved using Azure's audio-to-text transcription service, with Python for building the back-end models. The MLM model used to suggest related concepts is a pre-trained model from HuggingFace. While the RTMGE has been tested using the Zoom platform, the technology concept could be used in conjunction with other televideo platforms. The architecture supports plugging-in new procedures to analyze transcripts, which could be used to direct the interviewer to other kinds of suggestions, such as follow-up questions or to expand vague concepts. In this respect, the method is not limited to preference elicitation and could be deployed in requirements elicitation activities more generally.

5.4.3 Limitations and Challenges

The statistical significance tests did not demonstrate significance for our hypotheses, which means the observed differences could be due to random variation rather than a true effect of our real-time guided elicitation method. Several factors may have contributed to this lack of statistical power. One possible reason may be the small sample size we collected due to the time-consuming and costly process of recruiting and training qualified interviewers, recruiting interviewees, managing complex scheduling logistics, conducting the interviews, and performing detailed manual data analysis to ensure unbiased results, many of which required substantial manual efforts in order to derive unbiased and correct results. Another thing that could be done to improve statistical significance for our results is to use different statistics to test our hypotheses. We explored different statistical approaches to test our hypotheses, including linear mixed-effects models for continuous data and mixed-effect ordinal logistic regression for ordinal data, but both failed to reach significance. In contrast, we deliberately avoided simpler tests like the Ranked Sum test or Student's t-test without grouping, as these require data independence, an assumption violated in our study since multiple interviews conducted by the same interviewer are inherently dependent. The lack of statistical significance suggests that while our method shows promise, we cannot conclusively demonstrate that it provides genuine improvements over traditional interview approaches, and the observed benefits may be within the range of normal variation. Overall, this demonstrates several limitations of our method's capability of bettering requirements elicitation outcome from real-time interviews. We discuss them in more detail below.

The research design focuses on preference elicitation for directory services was applied to four directory service domains for which there are multiple web applications dedicated to each of these services: finding an apartment, finding a restaurant, finding a hiking trail and finding a clinic. There is no guarantee whether the same results may apply to other directory service domains, or to non-directory services, such as online banking or social media applications. That said, the fundamental technology used in the method is generalizable, which consists of an MLM pre-trained on general English corpus. One general limitation is that the suggestion results shall be less effective for less popular domains with limited appearance in the pre-training data. In such cases and domains, a common work-around is to fine-tune the base MLM to yield improved domain-specific embeddings that can be used to generate better suggestions. This can be done by continuing to train the pre-trained MLM with domain-specific text [180]. An alternative is to train MLM from scratch with a domain-specific corpus, which requires significantly more training data and computational power.

In addition, we also believe our method's implementation may be improved with more recent state-of-the-art approaches. With the recent advances in foundation and large language models (LLMs), in particular autoregressive training techniques for building language models like the Generative Pre-trained Transformers (GPTs), the capability of generating fluent and coherent natural language text has improved. Both our current interview transcription quality and related domain element suggestion may be improved with more recent state-of-the-art LLMs. Current interview transcription suffers from challenges such as forming grammatically correct transcription text and adapting to speaker accents to produce correct transcription. In our anecdotal evaluations of LLMs, we observe that they are resilient to these errors in transcription and that they could even be used to post-process transcriptions to improve both syntax and semantics. Other more advanced speech

recognition and audio transcription services may be used to replace the current choice to produce higher quality transcriptions. Besides, the MLM used in our back-end suggestion model may be replaced by an LLM that produces a higher frequency of domain-relevant terms, to further enhance our method's suggestion capability. LLMs may be used with prompt engineering techniques with few-shot or even zero-shot learning [97, 122, 204] to generate follow-up questions to be suggested to interviewers. LLMs may further aid the interview transcription analysis procedure. For example, LLMs can be used to summarize the transcript and generate the stakeholder preferences mentioned in an interview. This potential enhancement could reduce the need of the interviewer to filter which words identified by POS tagging are relevant or irrelevant.

Another aspect of the method that may be improved is to provide a visualizable page for the interviewer to track which concepts and which of their respective sub-concepts have been discussed in the interview. This will likely further ease the information overload on the interviewer's side and help them guide their interviews more easily and effectively, hence improving elicitation.

5.5 Threats to Validity

We now discuss threats to validity.

5.5.1 Construct Validity

Construct validity refers to whether we are measuring what we believe we are measuring [197]. In this study, we examine the concept of stakeholder preference. To define this concept, we conducted an early study on scenario authorship, in which we asked mobile app users to write scenarios about mobile apps that they frequently use and to describe their specific goals and preferences [80]. Based on this earlier study, we developed a definition of stakeholder preference from the evidence provided by stakeholders, including how these preferences are written syntactically. With this construct definition in mind, we manually identified the preference concepts from interview transcripts and carefully reviewed the identification process. The extracted preferences were independently cross-validated by two different people, and two people reached agreement on the results. When constructing and analyzing ontology as mentioned in Section 5.2.3, we chose to use the weighted average tree depth to perform the construction and analysis, with the assumption that concepts at a larger depth, which are more specific, shall have a larger weight in our statistical analysis. This linear weight assumes a direct, proportional relationship between depth and specificity, meaning that each level increase corresponds to a fixed increase in the weight. For example, the concepts at depth 4 is twice as specific as concepts at depth 2 in this formulation. We acknowledge that in reality, this may not always accurately quantify the concept specificity. With this in mind, we worked carefully on the ontology construction and ensured that the same schema is used across the control and the treatment group.

5.5.2 Internal Validity

Internal validity refers to the validity of the analysis and conclusions drawn from the data [197]. A few threats to internal validity are listed below.

Personality Effects

Personality effects consist of characteristics of the interviewer, such as race, age, gender identity, or demeanor, that may influence the interviewee's responses [105]. The interviewers each have different styles of interviewing, some may naturally be easier to elicit important information than others. Other aspects, such as the interviewer and interviewee's speech speed, or their natural abilities to build rapport with each other quickly, may also be factors affecting the elicitation results. To remedy this style bias, we randomly assign interviewers to the control and treatment groups and they remain in that group throughout the interviews. In addition, we trained all our interviewers on the basics of requirements elicitation interview techniques using the Ferrari et al. course on requirements elicitation interviews [61].

Learning Effects

Learning effects occur when interviewers learn from their professional trainings, experiences, or past interview experiences during the interview study period [101]. The interviewer may become more familiar with interviewing as they conduct more interviews, meaning that those interviews which occur in the later part of their timeline can produce better outcomes than the earlier interviews, independent of which group they were assigned to. Learning effects may not be as evident with interviewers who are experienced with interviewing, however. To that end, we trained the interviewers with requirement elicitation interview techniques using the Ferrari et al.'s course on requirements elicitation interviews [61], prior to the start of interviews to improve the interviewers' overall baseline experience. Moreover, the recruited interviewers all have completed at least one semester of requirements engineering coursework covering the interview elicitation techniques. We also adopted a randomized order for our eight interviewers in interview assignment, where all interviewers conducted their four rounds of interviews on different topics, and within each round, the interviewers of the same group interviewed on different domains, as shown in Table 5.1.

Fatigue and Boredom

The length of the interview and as well as the repetition of interviewing subjects on the same topic, could yield a worse outcome than the earlier interviews due to fatigue and boredom. To address this threat, we asked each interviewer to interview on four different domain topics and take breaks between interviews as needed with a recommendation of at least 15 minutes between interviews.

Hypothesis Bias

The interviewers and interviewees may be biased if they know the hypothesis, the research questions, or the purpose of our study. To reduce this threat, we hired and trained interviewers who were not researchers. The hired interviewers were not informed of our hypotheses, research questions, or study purpose. Furthermore, we avoided disclosing this information to the interviewees. Interviewees as participants in human subjects research were only told of the general purpose of the study to learn about the conduct of elicitation.

Analysis Bias

Analysis bias can arise if the analyst knew which study group (control or treatment) the unique preference-related concepts in an interview were attributed to. For example, they may unconsciously increase or decrease their effort for counts of one group over the other, or make assumptions about the performance of interviewer and interviewee in either of the two groups. To avoid this bias, we pooled the interview transcripts into a single group without attribution to the study group. The concepts extracted from each interview were further used to build the preference ontology tree without attribution to the study group. Hence, the analyst did not know which transcript belongs to which study group when they marked and counted entities.

Another source of bias could be the length of the interview, e.g., if interviews in one group were longer than in another group. To address this bias, we averaged the results by the number of turns in the interview, in the turn-based analysis method. To further avoid biases in the analysis, two people independently verified the results.

In addition, we have discussed why better elicitation results are observed from the treatment group. However, the causal relationships are mainly conjecture and cannot be tested. To remediate this aspect, we were careful in designing our randomized controlled experiments and running statistical tests on the results of the experiments in order to look for evidence to support our interpretations.

5.5.3 External Validity

External validity refers to the generalizability of the results [197]. Preferences are unique to individual stakeholders, and preferences elicited in interviews will vary based on interviewee personality and interests. To better support generalizability, we conducted our interview study in four domains of directory services and collected preference from 29 interviewees. We have also used eight interviewers, four of which used our methods in a treatment group, to broaden the study of our method's effects to different interviewer styles. However, the preference concepts elicited are only valid for this study, and the generalizability to other domains beyond the four domains studied 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 are an ideal fit to study individual stakeholder preferences. In addition, whether the effects we observed generalize to non-directory

service domains, or directory service domains we did not interview on, is unknown.

5.6 Summary

Requirements Elicitation in interviews is a challenging practice that may be made easier with real-time machine guidance, enabled by adopting various natural language processing techniques, including audio-to-text transcription, part-of-speech concept tagging, and MLM-assisted domain element suggestion. We present, in this chapter, a real-time machine-guided elimination method that incorporates the above techniques to guide requirements elicitation in interviews. Our results, however, failed to indicate statistical significance for demonstrating RTMGE's effectiveness in improving elicitation quality, with respect to the number of unique concepts elicited, and the concept specificity. We discuss the advantages and challenges associated with our method, and shed light on how future research may improve on our results.

Chapter 6

Requirements Elicitation Question Generation

Previously in the last chapter Chapter 5, we introduced methods to automatically generate and elicit domain elements using a BERT-based masked-language model (MLM). MLMs are limited to a specific format of input, namely a natural language input with a [MASK] token in it, and hence are limited to the task of mask filling. For example, previously in our work, we would ask the MLM model to fill in the mask for the input "I want an apartment that is [MASK].", where the MLM would produce texts such as "furnished", "nice", and "beautiful". On the other hand, LLM models like GPTs are more flexible in this aspect, in that it could accept various formats of natural language input and be asked to generate almost any format of output, not restricted to just mask filling tasks. Instead of generating just domain elements and leaving the formation of follow-up questions to interviewers, the flexibility of LLMs may enable us to move a step further and directly form meaningful follow-up questions. Hence in this chapter, we study the ability of LLMs to generate requirements elicitation interview questions, instead of just domain elements. Note that our design choice focuses on generating the questions but not the answers, because we believe obtaining real human stakeholder opinions and preferences is still essential and cannot be replaced by LLMs. Stakeholders have lived experiences with existing systems, understand unstated business rules, organizational politics, and historical context that even the most advanced LLMs cannot replicate without specific input. Moreover, LLMs lack many crucial aspects, such as the ability to validate requirements, the presence of emotional, moral and social dimensions [96], or the understanding of real-world constraints and accountability. In my opinion, the LLM serves more as a summarization of a majority of human inputs from its training data source [58], and should not completely replace all individual participants in requirements elicitation interviews.

6.1 Motivation and Goal

In requirements engineering, business analysts can employ interviews to elicit, acquire, identify, and elaborate requirements for information systems [202]. Interviews allow analysts to directly engage with stakeholders, who express their experiences and viewpoints, clarify ambiguity,

and provide context to their specific needs. By asking well-formulated questions, business analysts can probe topics with increased depth and uncover requirements that may not otherwise emerge. However, variations in domain knowledge, communication skill, and cultural differences, can all lead to different elicitation outcomes. Interviewers may find it stressful to process an interviewee's speech and identify appropriate follow-up questions during the interview, because of psychological hindrances such as excessive cognitive load. Good follow-up questions must be clear, relevant to the interviewee's speech, and informative to draw out tacit knowledge and hidden needs that may otherwise be ignored. Interviews are also inherently costly to conduct, and require significant time and effort to plan, schedule, and conduct, and to analyze responses [7, 36], which makes it difficult to scale interviews to large groups of stakeholders.

Advances in textual entailment have been transformative in generating high-quality human language [8, 51]. Since interviews involve natural language dialogue, both from interviewee speech and interviewer questions, this has made interviews a suitable beneficiary of these advances. In particular, autoregressively trained large language models (LLMs) like the Generative Pre-trained Transformers (GPTs) have been used to process and generate natural language for open-ended tasks where the structure of the output is not predetermined, including interview script generation [64], user simulation [4], transcript analysis [168], and chatbots for information elicitation [84]. These techniques are primarily focused on the preparation and post-processing stage of interviews, with relatively limited research studying elicitation at interview time.

In this chapter, we aim to study the effectiveness of LLMs in generating interview questions. We report the following work: 1) an empirical analysis of 14 interviews to identify the quantity of prior context needed for an interviewer to formulate a question; 2) experimental results comparing minimally guided GPT-4o-generated and human-authored follow-up questions for relevancy, clarity and informativeness; 3) a framework synthesized from prior work to outline common mistakes made by interviewers during elicitation; and 4) experimental results comparing GPT-4o-generated and human-authored follow-up questions to address common interviewer mistakes. The findings show that in general, GPT-4o generated questions are no worse or better than human-generated questions. However, when focusing on specific interviewer mistakes, GPT-4o-generated questions are rated more highly for relevancy, clarity and informativeness.

6.2 Research Questions and Methodologies

In this chapter, we investigate ways to generate follow-up questions for interviewers using textual entailment, in which the preceding interview context is used to entail the a good follow-up question using instruction-tuned LLMs. Our research starts with studying the types of follow-up questions in an interview and the amount of prior context needed for an interviewer to form a question, which aims to advance our knowledge of how follow-up questions are formed, as well as to what extent an interviewer needs to be aware of the whole interview when asking questions, while providing a dataset that we could use for our further analysis. Then we construct studies to learn how minimally guided LLM-generated follow-up questions compare to human-generated questions. Next, we provide a framework of common interviewer mistake types in asking follow-up questions, and use it to study to what extent can an LLM classify a mistake's presence, and

how LLM-generated questions compare to human-generated questions when provided with the mistakes to avoid. As part of our research design and evaluation, we introduce the following research questions (RQs):

RQ1: What are the types of follow-up questions in an interview and how many prior speaker turns are needed to formulate each question?

RQ2: How do minimally guided LLM-generated follow-up questions compare to human-generated questions?

RQ3: To what extent can an LLM classify whether a human-generated question demonstrates a common interviewer mistake type?

RQ4: When provided with common interviewer mistake types to avoid, how do LLM-generated questions compare to human-generated questions?

To address these questions, we first collect and conduct an analysis of interview transcripts to learn the follow-up question types and prior context to answer RQ1, and a literature review to identify common interviewer mistakes related to the formulation of interview follow-up questions, which will be used to answer RQ3 and RQ4. We then designed and conducted three studies: (1) a study to measure quality-related differences between questions generated by GPT-4o in comparison to questions raised by trained interviewers within the same interview context, which will provide answers to RQ2; and (2) a study to assess how well GPT-4o can detect whether interviewer-raised questions demonstrate any one of a number of common interviewer mistakes, which will answer RQ3; and (3) a study to measure the quality-related difference between GPT-4o generated questions and human analyst-authored questions, when both the LLM and human are instructed to avoid a common mistake, which shall answer RQ4. Below, we outline the data collection process for obtaining interviewer transcripts, followed by a detailed discussion of the experimental designs and evaluation methods.

6.2.1 Interview Transcript Collection

In this chapter, we reuse the collected control group interview transcripts from Chapter 5. The interviews were each randomly assigned to one of four directory service domains: apartment finding, restaurant finding, hiking trail finding, and clinic finding. The main focus of these interviews is to elicit stakeholder preferences.

6.2.2 Study 1: Minimally Guided Questions

RQ1 advances our knowledge by helping us to understand to what extent an interviewer needs to be aware of the whole interview when asking questions. Understanding the relationship between contextual awareness and question effectiveness contributes to our theoretical understanding of how requirements knowledge is constructed through dialogue. Moreover, the context length directly impacts the design of AI-assisted requirements elicitation methods. If interviewers need full context awareness to ask effective questions, then AI support systems must be designed to maintain and reference the complete interview history. Conversely, if effective questions can be generated with limited context, this enables more lightweight, real-time AI assistance that could be more widely adopted and computationally feasible.

The RQ1 is answered by manually reviewing the collected transcripts to identify all interviewer questions and to determine the minimum number of preceding conversational turns upon which a question depends. A question depends on a prior turn if the context provides information needed to comprehend the question. We define a **turn** as a single person's speech in the conversation from start to finish. For instance, if an interviewer poses a question and the interviewee responds, this exchange constitutes two turns. Our analysis reports turn frequency, which is the number of turns required to establish sufficient context for an interviewer question. This analysis also produced a dataset of 146 follow-up question contexts, mapping the preceding relevant turns to the corresponding interviewer questions. We refer to this dataset in the chapter as the *turn context dataset*, which will then be used in other experiments described in this chapter.

In addition to turn frequency, we used open coding [157] to classify the types of follow-up questions in the turn context dataset. To conduct this analysis, we first assigned a label to each question in the dataset based on their understanding of the relationship between the question and the prior dependent turns. For example, when an interviewer asks about two topics A and B in turn $n-2$, followed by the interviewee responding to only topic A in turn $n-1$, followed by the interviewer asking the interviewee to respond to topic B, the second follow-up question is labeled 'question probing' and its dependent on two prior turns. For each question in the dataset, we labeled the questions by inspecting the prior turns, reusing labels when appropriate and maintaining definitions for the labels. The coding process saturated when we had coded 32 questions, after which they discovered no new labels while coding the remaining questions. This process yielded a total of seven labels that we presented in Section 6.3.

We answer RQ2 and evaluate the GPT-4o-generated questions against the human-authored questions by relying on Paul Grice's pragmatic theory of productive conversations that includes four maxims: be relevant; be clear; be informative; and be truthful. Paul Grice was a philosopher of language who developed influential theories about how human communication works in practice. While truthfulness is important in requirements elicitation, particularly when interviewing an adversarial stakeholder, we chose in this research to assume that stakeholders are truthful. Thus, we adopt the remaining three maxims to evaluate questions, summarized and presented to the raters:

- *relevancy* - an interviewer question should be relevant to the topic or system and be situated within the flow of conversation
- *clarity* - an interviewer question should be clear and understandable by the stakeholder
- *informativeness* - an interviewer question should result in a stakeholder response that increases the quantity of information known about the system

Formally, we evaluate the following null hypotheses using an independent t-test:

H1.1.0: Minimally-guided LLM-generated questions are no different than human-generated questions with respect to relevancy.

H1.2.0: Minimally-guided LLM-generated questions are no different than human-generated questions with respect to clarity.

H1.3.0: Minimally-guided LLM-generated questions are no different than human-generated questions with respect to informativeness.

The alternative hypotheses are as follows:

H1.1.1: There is a significant difference between minimally-guided LLM-generated questions and human-generated questions with respect to relevancy.

H1.2.1: There is a significant difference between minimally-guided LLM-generated questions and human-generated questions with respect to clarity.

H1.3.1: There is a significant difference between minimally-guided LLM-generated questions and human-generated questions with respect to informativeness.

To test the hypotheses, we calculate the sample size required by the statistical tests, with medium effect size 0.5, power 0.8, and significance level $alpha = 0.05$. If the p-value we obtain is less than the significance level, then we reject the null hypothesis, otherwise the null hypothesis holds.

Specifically, the RQ2 and H1 (including all null and alternative hypotheses above) were answered and tested, respectively, by asking human's to judge the GPT-4o-generated and human-authored questions. To do so, first, we randomly sampled 20 questions from the turn context dataset. Next, we observed that the sampled questions and their prior turns, called an *instance*, contained speech artifacts and grammatical errors, likely due to audio transcription errors. To reduce the influence of grammar on human perception of question quality, we manually corrected the grammar of each sampled instance using Grammarly [1] before passing the instance as input to GPT-4o.

For each instance, we prompted GPT-4o to generate a follow-up question by only providing the prior turns and by omitting the interviewer's follow-up question. We prompt GPT-4o-2024-08-06, a state-of-the-art, closed-source, multi-modal model developed by OpenAI. The following prompt template shown in Figure 6.1 was used to generate the follow-up questions:

```
You are an AI agent capable of generating context summaries.
During a requirements elicitation interview with an interviewee
about how the interviewee conducts {interview domain},
the INTERVIEWEE and INTERVIEWER have had the following
conversation: {interview turns}. Generate a follow-up
question that the INTERVIEWER should ask next based on the
conversation. Restrict your response to only show the
follow-up question without explanation.
```

Figure 6.1: Prompt 1 to Generate Minimally Guided Questions

Since follow-up question generation is a creative task, we used the model's default temperature setting of **1.0** to promote non-deterministic outputs, which means the model could generate different paraphrases of the same question or even different questions from the same prompt.

This experiment consists of two groups: one group of survey respondents who only see GPT-4o-generated questions and one group who only sees human-authored questions. We designed two surveys per group with 10 questions per survey to reduce the overall time taken to complete the survey, while increasing the number of questions evaluated by each group to 20 questions.

[1] https://app.grammarly.com/

As previously stated, this sample size is calculated with medium effect size 0.5, power 0.8 and significance level $alpha = 0.05$. The instructions for each survey were the same.

Each survey consisted of ten question blocks, which were randomized for each participant to randomly distribute any ordering effects among questions. A question block consists of a one-sentence description of the domain, the transcript context preceding the question, the question, followed by three semantic scales for relevancy, clarity and informativeness as shown below, respectively:

- Please rate the question's relevance with respect to the interview conversation.
- Please rate the question's clarity.
- Please rate the question's ability to encourage the interviewee to provide an informative response.

Respondents choose a level on a 6-point scale from least to greatest, e.g., very irrelevant, irrelevant, somewhat irrelevant, somewhat relevant, relevant, very relevant. The other two scales were similarly labeled, replacing irrelevant/relevant with unclear/clear and uninformative/informative.

The surveys were published using the Qualtrics platform [2]. We recruited 32 survey participants to rate the questions in each survey (64 participants in each group) using the Prolific platform [3]. To enroll in the experiment, participants first signed an informed consent form, which includes information about their rights and confidentiality protection. In addition, the participants must be at least 18 years old, reside in the US at the time of the study, and speak English as their primary language. The participants were assigned to either the GPT-4o-generated or the human-authored question group, and participants were not informed about which group they were assigned to. Upon completion of the survey, each participant was paid $6 as compensation through Prolific.

Survey results were used to perform a two-tailed Student's T-test to test hypotheses H1, which we report in Section 6.3.

6.2.3 Synthesized Mistake Framework

We answer RQ3 and RQ4 by investigating the source of interviewer mistakes during interviews and by synthesizing a set of mistake criteria to inform how to generate a follow-up question. We first review the method used to synthesize these criteria, before describing two studies to answer the RQs.

Requirements elicitation research has considered the criteria that support formulating and asking good interview questions. We identified 14 published papers by keyword searching Google Scholar[4] across multiple fields, including requirements engineering, human-computer interaction, and software engineering. A paper was selected if it contains: a) criteria that a good interview question should meet; or b) mistake criteria that an interview question should avoid. If a mentioned criteria describe a good interview question without describing the mistake counterpart, we manually change the description to the mistake counterpart. For example, a good question should elicit only one kind of requirements each time [48], which can be restated as a mistake criterion

[2]https://www.qualtrics.com/
[3]https://www.prolific.com/
[4]https://scholar.google.com/

to "ask questions that involve multiple kinds of requirements." This process yielded 28 mistake criteria covering a range of concerns, including leading with the wrong interview opening, failing to construct the right interview ambiance, the wrong follow-up questions, the wrong way to frame a question, poor question flow, failing to meet elicitation goals and wrong closing of an interview. We acknowledge that our literature review process is not intended to be systematic to the point that it could cover all possible literature in this space of work.

From the 28 criteria, we down-selected to focus on 14 criteria that describe mistakes in two categories: 1) *follow-up questions*, which are questions asked as a follow-up to stakeholder's prior speech; and 2) *question framing*, which refers to how a question should be framed or formulated during the interview. We selected these two categories to focus our question generation efforts on because a large majority of requirements elicitation interview questions depend on stakeholder's prior speech in order to be formulated, and all questions need proper formulation, which means good follow-up questions and correct question framing are essential components to a successful elicitation interview. The mistake criteria for the selected categories are as follows. Each criterion is followed by one or more citations of papers which the criterion is derived from.

Follow-up questions:

- *Fail to elicit tacit assumptions*, i.e. fail to justify or authorize assumptions stakeholders tacitly make without justification. [150]
- *Fail to consider alternatives*, i.e., fail to look for alternative information or alternatives to existing requirements. [30, 150]
- *No clarification when unclear*, i.e., accepts what interviewee said without asking for clarification when words interviewee said are unclear. [48, 168]
- *No clarification when contradictory*, i.e., accepts what interviewee said without asking for clarification when words interviewee said are contradictory. [48, 168]
- *Fail to elicit tacit knowledge*, i.e., fail to elicit tacit knowledge known to interviewee but unknown to interviewer. [4, 16, 30, 48, 168]

Question framing:

- *Ask generic, domain-independent questions*, i.e., fail to ask questions related to the interview domain or the questions asked are too generic. [30, 64]
- *Ask questions too long or articulated*, i.e., ask questions too long or complicated that would likely require interviewee to ask for repeating or rephrasing multiple times. [30, 48, 61, 64]
- *Use jargon*, i.e., ask questions containing special words or expressions not in the common vocabulary and are difficult for interviewees to understand. [30, 65]
- *Ask technical questions*, i.e., ask questions that require technical knowledge in order to answer. [30, 61, 64]
- *Ask questions inappropriate to user's profile*, i.e., ask questions that cannot be answered by the interviewee given the interviewee's profile. [30]
- *Ask for solutions*, i.e., ask interviewee to present a solution to satisfy a requirement. [30, 64]
- *Ask questions that involve multiple kinds of requirements*, i.e., mix different categories of re-

quirements or multiple specific requirements within one category into a single question. [48]

- *Ask vague questions that lead to multiple interpretations*, i.e., ask questions that can be interpreted in more than one way. [30, 61, 64, 65, 84]
- *Ask vague questions which could infer no reasonable meaning*, i.e., ask questions that does not have enough context or clarity for interviewee to answer. [30, 61, 64, 65, 84]

With the mistake criteria framework, we intend to study mistake-guided question generation in two studies: first, to answer RQ3, a human analyst who was familiar with the mistake criteria but unaware of the method and results for generating follow-on questions was asked to classify whether the human-authored question demonstrates one of the 14 interviewer mistake types above and, if it does, then as part of the answer to RQ4, the human analyst is next asked to write a suitable question that avoids the mistake. In this study, we prompt GPT-4o to perform the same classification and question generation task over the same dataset, and then conduct a comparative evaluation of the GPT-4o and human analyst results. Second, we collect questions where both GPT-4o and the human analyst determine that the interviewer question demonstrates one of the mistake types and survey an online population to compare the questions, in order to answer RQ4.

6.2.4 Study 2: Mistake-Guided Question Classification

RQ3 is aimed to not only be a part of our study design as explained above, but also to help us validate the LLM's reliability as an evaluator - the LLM can then be used to generate questions to avoid mistake types, if it is capable of identifying mistake types in a way that aligns with human expertise. We answer RQ3 by comparing the classification results of GPT-4o and the human analyst. For this study we use GPT-4o-2024-08-06 with temperature set to 1.0. The human analyst in this study has three years of industry experience working as a RE researcher, and they were excluded from the research steps to prompt the model and review model output to avoid introducing bias into this evaluation. We now describe the details of our method.

With the transcripts collected in Section 6.2.1, we segmented the transcript into interviewee-interviewer pairs for each domain. Next, we labeled the pairs where the interviewer speech included a question, and removed all remaining pairs. Finally, we randomly sampled 30 pairs and asked the analyst to separately classify whether the interviewer question demonstrates the fourteen mistake criteria shown in Section 6.2.3. For 30 questions pairs, this yields $30 \times 14 = 420$ classifications. To prompt GPT-4o to perform the same task, we designed a prompt template to be used once for each question-pair and criterion, which is shown in Figure 6.2:

The {`domain keyword`} is one of apartment, restaurant, trail, clinic. In addition, when designing the prompts for mistake criteria "No clarification when contradictory," "Ask technical questions," "Ask questions inappropriate to user's profile," and "Ask for solutions," we add a one-shot example to illustrate the meaning of the criterion in order to improve performance. We discuss insights from the prompt tuning process in Section 6.4.1.

Whenever the human analyst identified that a question demonstrates a mistake type, the human analyst is asked to proceed to generate a suitable question that avoids the mistake. To fully complement the human analyst data, we designed a second prompt to generate questions that avoid the mistake type identified by GPT-4o. The following prompt template shown in Figure 6.3 was used

```
You are an AI agent capable of conducting requirements
elicitation interviews. During a requirements elicitation
interview with an interviewee about how the interviewee
conducts {domain keyword}, the INTERVIEWEE said '{interviewee
speech}'. Then the INTERVIEWER asked a follow up question
by saying '{interviewer question}'. Standard: {mistake
criterion}. Please classify based solely on whether the
INTERVIEWER's response meets this specific standard, and
refrain from using any other standards related to follow up
questions when you classify. If the INTERVIEWER's response
meets this standard, output 'Yes', otherwise output 'No'.
Restrict your response to output only 'Yes' or 'No' without
explanations.
```

Figure 6.2: Prompt 2 to Classify Follow-up Questions

to collect these questions.

```
You are an AI agent capable of conducting requirements
elicitation interviews. During a requirements elicitation
interview with an interviewee about how the interviewee
conducts {domain keyword}, the INTERVIEWEE said '{interviewee
speech}'. Generate a follow-up question that meets the
following criterion based ONLY on what the INTERVIEWEE said,
and restrict your response to only show the follow-up question
without explanation. Criterion: {mistake criterion}
```

Figure 6.3: Prompt 3 to Generate Mistake-Guided Questions

To answer RQ3, we compare the GPT-4o classification results against the human analyst classification results for the 420 classification instances. Ahmed et al. (2025) argue that LLM-based labeling of software engineering artifacts is approaching human-level performance for some classification tasks, including code summarization, causality and semantic similarity [6]. In these tasks, differences of human opinion are measured using inter-rater reliability. Building on this prior work, the RQ3 aims to assess the relative performance of LLM-based classification by calculating an agreement rate, which is the percentage of instances that the GPT-4o classification agrees with that of the human analyst, along with specific details of how the agreement rate is distributed across different mistake types, and report the results in Section 6.3.2. We also report the Cohen's Kappa between the GPT-4o and the human analyst classifications.

6.2.5 Study 3: Mistake-Guided Question Generation

After collecting all GPT-4o-generated and human analyst-authored questions, we construct a second dataset that consists of the 30 records containing the interviewee speech, interviewer question, domain keyword, and mistake criterion identified by the analyst. Next, we duplicate this data into two sets: one with the interviewer question replaced by the GPT-4o-generated question, and one with the interviewer question replaced by the human analyst question for the given criterion. This yielded a dataset consisting of 128 total records, which exceeds the number of records required by a power analysis with medium effect size 0.5, power 0.8 and significance level 0.05. Next, we randomly shuffle the combined dataset, place the 128 records into 32 surveys, wherein each survey describes four follow-on questions.

The 32 surveys each contain four pairs of one GPT-4o-generated question and one human analyst-authored question, including the same interviewee speech, domain keyword, and mistake criterion assigned to the pair. The source of the questions (GPT-4o or human analyst) was not indicated and both the question pairs and the order of the two questions in each pair were randomized. The participants were asked to choose which of the two questions better avoids the mistake explained in the mistake criterion. Note that the human analyst is expected to write a question that avoids a given mistake to the best of his knowledge, which does not mean he is expected to always be able to avoid the given mistake. In addition, participants were asked to evaluate the questions for relevancy, clarity, and informativeness using a 5-point semantic scale from least to greatest, e.g., very irrelevant, somewhat irrelevant, neutral, somewhat relevant, very relevant.

The survey was published using Qualtrics, and we recruited 32 participant from the Prolific platform, wherein each participant was randomly assigned to one of the surveys. Prior to enrolling in the study, each participant must complete the informed consent form describing their rights and confidentiality protection. In addition, the participants must be at least 18 years old, currently reside in the US, and speak English as their primary language. Upon completion of the survey, each participant is paid $6 as compensation through Prolific.

We test the following null hypotheses:

H2.1.0: When provided with 14 types of interviewer mistakes to avoid, LLM-generated questions are no different from human-generated questions with respect to avoidance of mistakes.

H2.2.0: When provided with 14 types of interviewer mistakes to avoid, LLM-generated questions are no different from human-generated questions with respect to avoidance of relevancy.

H2.3.0: When provided with 14 types of interviewer mistakes to avoid, LLM-generated questions are no different from human-generated questions with respect to clarity.

H2.4.0: When provided with 14 types of interviewer mistakes to avoid, LLM-generated questions are no different from human-generated questions with respect to informativeness.

The alternative hypotheses are:

H2.1.1: When provided with 14 types of interviewer mistakes to avoid, there is a significant difference between LLM-generated questions and human-generated questions with respect to avoidance of mistakes.

H2.2.1: When provided with 14 types of interviewer mistakes to avoid, there is a significant difference between LLM-generated questions and human-generated questions with respect to relevancy.

H2.3.1: When provided with 14 types of interviewer mistakes to avoid, there is a significant difference between LLM-generated questions and human-generated questions with respect to clarity.

H2.4.1: When provided with 14 types of interviewer mistakes to avoid, there is a significant difference between LLM-generated questions and human-generated questions with respect to informativeness.

Again as in Study 1, we calculate the sample size required by the statistical tests, with medium effect size 0.5, power 0.8, and significance level alpha = 0.05. If the p-value we derive is less than the significance level, then we reject the null hypothesis, otherwise the null hypothesis holds.

We evaluate the ability to avoid the mistake types using a mixed-effect Bradley-Terry model, which are preferred in paired comparison experiments. The Bradley-Terry model is a generalization of a binomial model that can be used to predict the preference between two items in a paired comparison [28]. We chose to use the mixed-effect Bradley-Terry model instead of the standard model because each survey taker in our survey study rated four question pairs, which means the responses are not independent. A mixed-effect Bradley-Terry model accounts for this dependency by incorporating random effects for each rater, allowing us to model individual differences in rating tendencies, and thus mitigates bias introduced by variance in individual perceptions and ensures a more reliable estimation of the relative quality of the questions. Further, the mixed-effect model allows us to generalize findings beyond the specific raters in our study, making the results more robust and reflective of the actual question quality.

We evaluate relevancy, clarity, and informativeness scores provided by survey respondents using a mixed-effect ordinal logistic regression model. This model deals with ordinal data and allows us to have fixed effect for the question source, and random effects for each rater, similar to the mixed-effect Bradley-Terry model.

6.3 Results

We now report our results for each study described in Section 6.2.

6.3.1 Minimally Guided Question Generation Results

In Section 6.2.2, RQ1 asks "What are the types of follow-up questions in an interview and how many prior speaker turns are needed to formulate each question?" Figure 6.4 reports the speaker turn distribution for number of turns required by each follow-up question. On an average, the turn context size needed to formulate a question is one, i.e., providing the interviewee's last response was sufficient to formulate a follow-up question. We also note that 71/146 follow-up questions require no context, 104/146 require *up to one speaker turn*, and 98% (144/146) of questions require *up to four turns*.

During our analysis of the required speaker context, we open-coded the interviewer's follow-up questions to infer the interviewer's possible motivation behind raising the question. The following seven-question typology was derived from the context and aims to describe this possible motivation:

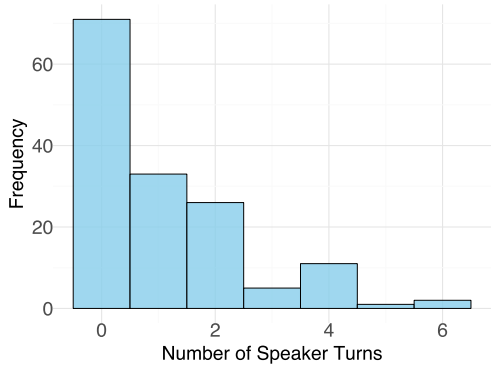

Figure 6.4: Speaker Turn Distribution Required By Follow-up Questions

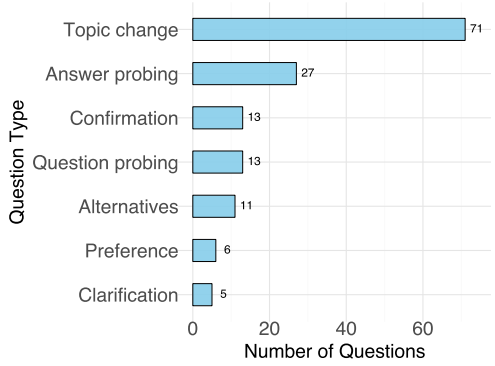

Figure 6.5: Follow-up Question Type Distribution

- **Topic change**: Questions in which the interviewer changes the conversation topic to talk about a different and unrelated topic than the one represented by the prior turns.
- **Answer probing**: Questions in which the interviewer appears to further probe a concept that the interviewee mentioned in their last turn.
- **Confirmation**: Questions in which the interviewer repeats or paraphrases the interviewee response to ensure that they correctly understand the interviewee's statement.
- **Question probing**: Questions in which the interviewer asks about a concept within the current topic scope but that was missing from the interviewee's last turn(s).
- **Alternative-seeking**: Questions in which the interviewer broadly asks about alternatives to a concept being discussed (e.g., what else questions).
- **Preference-seeking**: Questions in which the interviewer expects a yes/no answer from the interviewee to accept or reject an otherwise provisional interviewee preference.
- **Clarification**: Questions in which the interviewer asks about an ambiguous or vague concept in the interviewee's last turn.

Figure 6.5 presents the follow-up question typology distribution. When we review this distribution by question type, we observed that the number of turns needed as context varies. Topic change

questions require no prior speaker turns, whereas answer probing, confirmation, and clarification questions are primarily formulated using one speaker turn. Question probing, alternative-seeking, and preference-seeking questions require at least two prior turns.

The RQ2 asks "How do minimally guided LLM-generated follow-up questions compare to human-generated questions?" To assess performance differences between GPT-4o-generated and human-authored questions, we conducted a two-tailed Student's T-test for relevancy, clarity, and informativeness (see Table 6.1). The results show no statistically significant differences between the two groups for any metric for $p \leq 0.05$. Hence the null hypotheses H1.1.0, H1.2.0 and H1.3.0 are held.

Table 6.1: Study 1 Results: Human vs GPT-4o

Metric	**Informativeness**	**Relevancy**	**Clarity**
Human Avg Score	4.6	4.8	4.9
GPT-4o Avg Score	4.8	5.0	5.1
p-value	0.17	0.08	0.10

6.3.2 Mistake Types Classification Results

In Section 6.2.4, we described our study to answer RQ3, which asks "To what extent can an LLM classify whether a human-generated question demonstrates a common interviewer mistake type?"

We answer RQ3 by comparing GPT-4o and human classifications of whether an interviewer question demonstrates one of 14 mistake types. A total of 420 classification were separately obtained from GPT-4o and a human analyst. We observe that GPT-4o and the human analyst agree 81.0% (340/420) of the time, with a Cohen's Kappa equivalent to 0.61, which indicates substantial agreement [126]. Specifically, the human analyst classified 177/420 while GPT-4o classified 163/420 instances as having demonstrated a given mistake type. The classification distribution is presented in Table 6.2, where the first column is the mistake type, the second and third columns are the human's and GPT-4o's classification count for demonstrating the mistake type, respectively, and the last column is the agreement rate.

Finally, we observed 128 questions where both GPT-4o and the human analyst classified a given mistake type. These instances are further analyzed and described in Section 6.2.5.

6.3.3 Mistake-guided Question Generation Results

In Section 6.2.5, RQ4 asks "When provided with common interviewer mistake types to avoid, how do LLM-generated questions compare to human-generated questions?" To answer RQ4, survey respondents were presented with one turn of interviewee speech, with a question pair consisting of one interviewer question generated by GPT-4o and one authored by the human analyst, and a mistake type. The respondent was asked to select the question that best avoids the mistake type.

Table 6.2: Classification Results For Each Mistake Type

Mistake Type	Human	GPT-4o	Agreement Rate
Fail to elicit tacit assumptions	19	25	66.7%
Fail to consider alternatives	30	28	93.3%
No clarification when unclear	20	16	60.0%
No clarification when contradictory	7	2	83.3%
Fail to elicit tacit knowledge	21	20	70.0%
Ask generic, domain-independent questions	7	10	90.0%
Ask questions too long or articulated	11	7	86.7%
Use jargon	7	1	80.0%
Ask technical questions	5	1	86.7%
Ask questions inappropriate to user's profile	5	0	83.3%
Ask for solutions	6	2	86.7%
Ask questions that involve multiple kinds of requirements	7	12	76.7%
Ask vague questions that lead to multiple interpretations	27	27	93.3%
Ask vague questions which could infer no reasonable meaning	5	12	76.7%
Total	177	163	81.0%

Out of the 128 questions pairs, the GPT-4o-generated questions were selected 87 times, while the human questions were selected 41 times, indicating respondents believed that GPT-4o was more proficient at addressing the mistake type in 68.0% of question pairs.

The mixed-effect Bradley-Terry model reported a p-value $4.23 \times 10^{-8} \leq 0.05$, in which case we reject the null hypotheses H2.x.0 that there is no difference between GPT-4o and the human analyst. The probability that GPT-4o generates a better question that avoids a given mistake type is about $2.662/(1 + 2.662) = 93.5\%$ based on the odds ratio of 2.662.

Survey respondents were asked to rate on an ordinal scale of 1 to 5 for how relevant, clear, informative each question is. A mixed-effects ordinal logistic regression model was used to analyze the scores, which is reported in Table 6.3, including the GPT-4o and human average score, question win rate for relevancy, clarity, informativeness, the tie rate, and the p-value.

Among the 128 question pairs, GPT-4o scored better on average than the human-authored questions with respect to all three quality criteria, and the p-values are all less than 0.05.

Table 6.3: Relevancy, Clarity and Informativeness Results

Metric	Relevancy	Clarity	Informativeness
Human Avg Score	3.5	3.9	3.6
GPT-4o Avg Score	4.4	4.5	4.1
Human Win Rate	21.1%	17.2%	25.8%
GPT-4o Win Rate	59.4%	42.2%	47.7%
Tie Rate	19.5%	40.6%	26.6%
p-value	1.21×10^{-10}	1.34×10^{-6}	1.77×10^{-5}

6.4 Discussion

We now discuss our prompt design insights, context in question generation, question quality, and anecdotal results from a side study of GPT-4o's efficacy to generate follow-up questions that simultaneously avoid multiple mistake types.

6.4.1 Prompts Design

We designed the GPT-4o prompts using persona-based prompting technique [188], consisting of the system prompt "You are an AI agent capable of conducting requirements elicitation interviews."

When designing the prompts, our tuning and experimentation process revealed that GPT-4o failed to distinguish between the interviewer and interviewee speech. This challenge manifests as a classification error, wherein the model incorrectly attributes speech or misinterprets the question intent. We found that a simple typographical modification by capitalizing the role identifiers "INTERVIEWER" and "INTERVIEWEE" significantly reduced classification error. This finding aligns with prior research, which reports that LLM performance is sensitive to separators and capitalization changes [162].

In addition, because LLM performance is shown to be poor in the presence of negation [103], we reformulated the mistake description from a negative to a positive framing. For example, we included the instruction that "a good follow-up question should consider alternatives" instead of "a good follow-up question should not fail to consider alternatives." We found that 4/14 mistake types produced notably lower performance, for these we added a one-shot demonstration as recommended by prior work [24]. To illustrate, we added "For example, it's inappropriate to ask users about how to design a specific feature, or what would an ideal user interface look like." to clarify the meaning of the "ask for solutions" mistake type. For 9/14 mistake types in the classification prompts, we added step-by-step instruction. For example, in the "no clarification when contradictory" mistake type, we added "To classify whether the INTERVIEWER's question meets this standard, first consider if the INTERVIEWEE mentioned anything contradictory. If it does not, then the standard is met. Otherwise, look at whether the INTERVIEWER's question tries to clarify the contradiction."

6.4.2 Context in Question Generation

In our analysis of the number of turns or context required to ask a follow-up question, we observed that 70% of follow-up questions require zero or one prior speaker turns, which suggests that interviewers rely primarily on the most recent speech when formulating questions. This observation indicates that interviewers may lack capacity to generate questions from longer context windows due to psychological processes, such as cognitive load. This has two consequences: (1) LLM-supported tools may effectively generate human-comparable follow-up questions by focusing on the immediate context, reducing the need for extensive conversational history. In fact, a context window of four prior speaker turns appears sufficient to account for 98% of all follow-up questions observed. In addition, (2) LLM-supported tools could provide above-human performance if they can enable human interviewers to track information across larger context windows when generating follow-up questions.

The follow-up question typology yields practical implications. For instance, recognizing that topic change questions typically require no prior context could help systems initiate new conversational directions efficiently. Meanwhile, understanding that answer probing, confirmation, and clarification questions are closely tied to the most recent turn could guide models in prioritizing recent content when formulating these question types, whereas for question probing, alternative-seeking, and preference-seeking questions that demand a longer context.

6.4.3 Question Quality

In this work, we conducted a minimally-guided question study that shows that GPT-4o achieves comparable performance to human interviewers when generating follow-up questions. The mistake-guided question study shows that GPT-4o generates better questions than a human analyst with regard to relevancy, clarity and informativeness. The improvement may be attributed to how the mistake types naturally align with the quality criteria of relevancy, clarity, and informativeness. For example, avoiding the mistake "fail to consider alternatives," may yield questions with better informativeness, or avoiding the three mistakes "ask questions too long or articulated," "ask vague questions that lead to multiple interpretations," and "ask vague questions which could infer no reasonable meaning" may yield better clarity. Finally, avoiding the two mistakes "Ask generic, domain-independent questions," and "ask questions inappropriate to user's profile" may yield more relevant questions. By guiding the LLM to avoid these mistake types, the LLM may generate questions that are more performant on the evaluation criteria. If true, then research to better understand the underlying causes behind failure to perform elicitation in other ways (i.e., the underlying mistakes) could improve LLM instructional design to generate better interview questions.

6.4.4 Side Study: Simultaneous Avoidance of Mistakes

In Study 3, we investigate how well an LLM can generate questions to avoid a single mistake type. We conducted a side study to preview how well LLM-generated questions can simultaneously avoid multiple interviewer mistake types. This research question is interesting because simultaneous avoidance could yield questions of even better quality. However, prior work shows that LLMs

are less performant when multiple constraints must be conjointly satisfied in a single task [69,184]. When prompting GPT-4o to simultaneously avoid all fourteen interviewer mistake types listed in Section 6.2.3, and to evaluate whether a GPT-4o-generated question successfully avoids each mistake type using the same 30 interviewee-interviewer pairs from Study 3, we observed that 66/420 classifications still demonstrate a given mistake type. Table 6.4 presents the success-avoidance rate for each mistake type. Besides, 1/30 questions successfully avoided all mistake types, and 28/30 questions successfully avoided at least 11 mistake types. This shows that while an LLM may avoid a majority of mistake types simultaneously, more work is needed to evaluate and improve the avoidance rate. We did not, however, evaluate how these generated questions compare to the human-authored questions using the same Study 3 protocol. We postpone that question to future work.

Table 6.4: Classification Results For Each Mistake Type

Mistake Type	**Avoidance Rate**
Fail to elicit tacit assumptions	76.7%
Fail to consider alternatives	26.7%
No clarification when unclear	70.0%
No clarification when contradictory	83.3%
Fail to elicit tacit knowledge	90.0%
Ask generic, domain-independent questions	96.7%
Ask questions too long or articulated	90.0%
Use jargon	100.0%
Ask technical questions	100.0%
Ask questions inappropriate to user's profile	100.0%
Ask for solutions	100.0%
Ask questions that involve multiple kinds of requirements	60.0%
Ask vague questions that lead to multiple interpretations	93.3%
Ask vague questions which could infer no reasonable meaning	93.3%

6.5 Threats to Validity

Construct validity refers to whether we are measuring what we believe we are measuring [197]. In our research, the interview contexts used were obtained from real interviews conducted by trained interviewers. The questions in Study 1 were obtained from these transcripts. In Study 2 and 3, the contexts were reused, however, the questions were authored by a human analyst who was not engaged in the interview and who had access to the mistake catalog during question formulation. In addition, question quality is difficult to measure. To address this threat, we adopted the Paul Grice's pragmatic theory of productive conversations, specifically, relevancy, clarity, and

informativeness to measure quality. Moreover, the interviewer mistake types were synthesized from literature review, which speaks to their broad observation across multiple studies, but which may not comprehensively cover all possible mistakes.

Internal validity refers to validity of analyses and conclusions drawn from the data [197]. In our evaluation, the survey participants' pre-existing biases about AI-generated content could influenced their judgments. To reduce this threat, we randomly shuffled questions and blinded raters to the source of questions (human or GPT-4o). For example, in Study 3, the question source was blinded and the order of the questions was randomized to further evenly distribute any impact of human raters being biased toward the first or last shown question. The human analyst who classified interviewer mistakes and generated alternative questions may also have pre-existing biases about AI-generated content. We mitigated this threat by preventing the human analyst from learning about the LLM generation procedures and by concealing the survey designs and hypotheses from the analyst. In addition, humans may encounter cognitive challenges, such as fatigue and attentional limitations, when tasked with classifying interviewer mistakes and generating questions that avoid the mistakes as part of the research. This may produce questions perceived as less relevant, clear or informative than if a human performed the same task without cognitive challenges. In contrast, LLMs are prone to hallucination, i.e. generating fluent but incorrect or misleading content [**?**]. This means that some GPT-4o-generated questions may appear professionally crafted or linguistically advanced, while omitting necessary domain-specific information, or they do not adequately advance the elicitation objectives. Our evaluation, which relies on specific quality assessments, does not specifically detect hallucinations, however, if hallucinations correlate with quality decrement, then the quality measurements would be impacted. In addition, the open coding and analysis of the interview transcripts to answer RQ1 were performed by one person, without formal validation measures. Systematic bias in the question categorization may exist, which can affect the reliability of the results.

External validity refers to the generalizability of results [197]. This chapter mainly focused on requirements elicitation interviews in four directory service domains and on eliciting stakeholder preferences, hence the generalization of our results to other types of requirements elicitation interviews is unknown. While internet users spend large amounts of time searching for information online and thus observations from interviews about directory services could generalize to other Internet applications, these interviews did not cover subjects with safety-critical or performance requirements, such as autonomous driving or medical device software. Besides, preference elicitation is a specific variant of interviewing that may not be representative of all types of interviews. Hence our analysis that relied on the collected interview transcripts may not adequately generalize to all types of requirements elicitation interviews.

The interview transcripts were created by interviewers with at least two years of software engineering experience who were trained to conduct interviews using the Ferrari et al's techniques [61]. While these interviewers are former professionals returning for graduate study, however, the interviews were conducted in an academic setting and thus these results may not extend to industrial settings. In addition, while the interviewees were required to have domain expertise, the interviewers were not required to have this expertise and were not assigned the job role to interview users for an app they were developing. These differences mean that the human performance observed in

response to RQ2 may not reflect the capabilities of expert practitioners, thus overstating the relative performance of LLM-generated questions. Finally, while the interviewees are all users of apps in the category for which they were recruited to be interviewed, they may not be representative of the population of users for any specific app.

The evaluators of the LLM- and human-generated questions were recruited using the Prolific platform. Thus, the question quality measurements reflect the opinions of Internet users and they were not individuals trained in discourse analysis. As a result, the evaluators may interpret the evaluation criteria (relevance, clarity and informativeness) differently from trained analysts when rating or comparing follow-up questions and thus the average ratings could be different from those analysts. According to Douglas et al., in comparison to the US population, the Prolific population is 67.5% female, skews younger, and is similar in age and income [47]. We limited respondents to a US location and native English speakers.

We used GPT-4o as the LLM in our studies, and results may not generalize to other language models. The rapid pace of LLM development means that newer models may demonstrate behaviors different from those we observed here. The prompts we used for our task may also not transfer other models.

Our approach studied how well GPT-4o generates follow-up interview questions that traditionally require significant human expertise. However, we did not assume, nor demonstrate that GPT-4o can perform the full range of interviewing capabilities, such as building rapport with interviewees or adapting to cultural differences. Our evaluations focused narrowly on question generation quality rather than comprehensive interviewing competence. Moreover, we did not empirically study whether our methods could address the cost and scaling challenges of traditional elicitation interviews.

6.6 Summary

In this chapter, we proposed a framework outlining common interviewer mistake types and for evaluating follow-up question quality. We collected interview data and conducted studies to test the capability of GPT-4o in classifying whether a question demonstrates a mistake type, and to evaluate LLM-generated questions against human-authored questions in controlled experiments. Our findings demonstrate that minimally-guided LLM-generated questions are not statistically better or worse than human-authored questions with respect to clarity, relevancy and informativeness. Moreover, LLM-generated questions outperformed human-authored questions in mistake-guided question generation. We believe our findings indicate that LLMs can be employed to support requirements elicitation interviews with improved outcomes.

Although we did not fully incorporate the question generation process into interviews, we believe that it is possible to do so by collecting interview transcripts in real-time and presenting up to four speaker turns to the model with prompts aimed at generating mistake-avoiding questions. Presenting these questions to the interviewer for their review and possible selection could yield more performant interviews. Similarly, the mistake classification study described in this chapter would potentially transform into a tool to help interviewers recognize their mistakes during interviews. In future work, we envision studying this performance in the context of variable communication

skills and cultural differences, as well as excessive cognitive load.

Other sophisticated techniques for using the LLM to generate interview questions have not been studied in this work. For example, it is possible to create and fine-tune the LLM on an interviewer-interviewee conversation dataset, apply Reinforcement Learning from Human Feedback (RLHF) on interview quality using human preferences [143], use Retrieval-Augmented Generation (RAG) to automatically retrieve relevant and prior interview excerpts to inform follow-up questions [120], use multi-agent approaches to create reflective loops wherein one LLM evaluates another LLM's questions [203], or apply domain-specific additional pretraining [193], if adapting the approach to relatively rare domains. We leave the exploration of such techniques to future work.

Chapter 7

Contributions and Final Remarks

In this chapter, I discuss my thesis' main contributions, my work's implications for requirements engineering, limitations and future work, and lastly the open questions remaining in this field that I believe would be particularly interesting to research on.

7.1 Main Contributions

This thesis contained extensive work to use embeddings-assisted methods to improve some requirements extraction and elicitation practices, particularly for better domain knowledge modeling, and for refined requirement artifacts, in our case interviews, via guided elicitation with domain knowledge we extracted, or via interview questions that we generate automatically. The main contributions are: 1) we generalize typed dependency techniques to create domain models for directory service software; 2) we use a wider range of embeddings-assisted methods and natural language techniques to model domain knowledge both from scenarios and from word embeddings; 3) we use domain knowledge to refine interview as an requirement artifact for better elicitation; and 4) we use an instruction-tuned LLM that encapsulates domain knowledge in its training process to generate high-quality interview questions for elicitation interviews. The contributions of these work together serve to 1) bridge the gap between the new emerging technologies and traditional requirements acquisition practices; 2) bring to sight how embeddings-assisted methods could improve CrowdRE, domain modeling, and more broadly general user-oriented non-functional requirements acquisition; 3) focus on previously more overlooked area of direct elicitation from users instead of via user-system interaction's telemetry data; 4) focus on previously more overlooked user preferences tacitly embedded in natural language text and conversations they produce; 5) use embeddings-assisted methods to resolve some challenges previously associated with interview elicitation; and 6) shed light on a future of automated, and better, requirement acquisition. The specific contributions for domain knowledge modeling and for guided elicitation in interviews are shown below. We discuss the detailed contribution of each work below, including their novelty, importance, and how they may have advanced our knowledge towards more automated requirements extraction and elicitation.

7.1.1 Domain Knowledge Modeling

We proposed two approaches to extract domain models from a user-authored scenario corpus and to extract domain models from word embeddings using masked language models (MLM).

In the first approach, the scenario corpus we constructed, and the pipeline to extract stakeholder preferences from them, provide greater understanding of preferences in scenarios. Developers can enhance software with features that better satisfy such preferences. Furthermore, based on our extraction 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.

This work is novel because, to our knowledge, it's the first end-to-end method to extract stakeholder preferences from scenarios. Our dataset and supporting tools, including a trained transformer model that developers may use, are available online [205].

Our preference-based elicitation method complements traditional elicitation approaches, such as personas [74], app reviews and issues [131]. 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.

In the second approach, our method of domain elements extraction from word embeddings using masked language models (MLM) provide developers a way to enhance an existing domain model with supplementary semantic knowledge embedded within word embeddings. The data and code are available online [206].

7.1.2 Guided Elicitation in Interviews

Our real-time machine-guided elicitation (RTMGE) method and its tool improves and partially automates the process of preference elicitation in interviews. To our knowledge, we are the first to attempt at creating a real-time interview elicitation method to guide interviewers during interviews, in order to improve elicitation quality.

Our method that supports real-time guided stakeholder preference elicitation in interviews enables developers and requirement analysts to better extract stakeholder preferences during interviews, likely resulting in more concepts elicited per interview, and more in-depth preferences explored. Requirement analysts can use our tool or create a tool with similar structures to conduct preference elicitation interviews, and use the elicited stakeholder preferences to enhance their products.

Our method also aids in tackling various challenges associated with interview elicitation practices. Specifically, the suggestion functionality may enhance the interviewer's domain knowledge and help them come up with appropriate follow-up questions more quickly and easily. This in turn enable interviewers to guide interviewees in probing deeper into a topic of interest during an interview. We also add real-time interview transcription functionality into the tool, and use part-of-speech tagging to highlight potential concepts of interest in the transcription, to help interviewers more easily visualize potential topics of interest and generate suggestions, and to reduce the possibility of the interviewer going off-topic for a long time and forgetting other previously mentioned topics. The suggestions along with the transcription may also ease the cognitive load of the interviewer and help them manage the overwhelmingly rich information in the interviews.

Our second piece of work in this area, the requirements elicitation question generation work, proposed a framework outlining common interviewer mistake types and for evaluating follow-up question quality. Moreover, We collected interview data and conducted studies to test the capability of GPT-4o in classifying whether a question demonstrates a mistake type, and to evaluate LLM-generated questions against human-authored questions in controlled experiments. Our findings demonstrate that minimally-guided LLM-generated questions are no better or worse than human-authored questions with respect to clarity, relevancy and informativeness. Moreover, LLM-generated questions outperformed human-authored questions in mistake-guided question generation. We believe our findings indicate that LLMs can be employed to support requirements elicitation interviews with improved outcomes. Although we did not fully incorporate the question generation process into interviews, we believe that it is possible to do so by collecting interview transcripts in real-time and presenting up to four speaker turns to the model with prompts aimed at generating mistake-avoiding questions. Presenting these questions to the interviewer for their review and possible selection could yield more performant interviews. In future work, we envision studying this performance in the context of variable communication skills and cultural differences, as well as excessive cognitive load.

7.2 Implications for Requirements Engineering

The thesis contributes to the field of requirements engineering in the areas of domain knowledge modeling and requirements elicitation. When viewed within the general context of requirements engineering, these contributions address several ongoing challenges and open opportunities for future research directions.

7.2.1 Towards Automation and AI-supported Methods

As software development processes become more agile and are now subject to often shorter deployment deadlines, there is a growing need for automated techniques to accelerate requirements elicitation and analysis, and to reduce human effort in the process. The thesis work aligns with this trend by studying embeddings-assisted automated methods that emphasize productivity in elicitation by modeling domain knowledge to produce domain concepts and elements, and by guiding elicitation interviews with domain knowledge and with question generation. For example, the work on domain knowledge modeling from scenarios described a method that could identify preferences from collected scenarios in an effective way, and the work on interview refinement saves human effort by suggesting related domain elements during real-time interview elicitation, or by utilizing large language models to generate effective interview questions of good quality to avoid or reduce the number of interview mistakes.

Moreover, the automated methods may be used to supplement traditional methods. For example, significant work in requirements engineering has shown that a major challenge in domain knowledge modeling lies in the difficulty of acquiring the tacit knowledge that stakeholders may possess but that may be hard to articulate [16, 63]. My work on embeddings-assisted methods, contrary to prior work that leverages traditional techniques, such as rule-based extraction [12], value-oriented elicitation [49], requirements catalogue construction [59], and more [127, 170, 176], to solve this problem, employs recent advances in natural language processing (NLP) and embeddings-assisted techniques to supplement domain knowledge modeling and to guide elicitation practices. In my view, both approaches are effective, traditional or contemporary, and this work bridges the gap between the two approaches so that my method could supplement other methods to create a more systematic approach that may result in a higher quality elicitation. My work on domain knowledge modeling aims to address limitations in human knowledge that often exist in traditional approaches, which can impact the completeness of domain models extracted, by shifting the knowledge source to trained machine learning models to support identifying gaps in domain models. Although these models are trained based on human knowledge, they provide a place where knowledge is gathered and stored in a collective way that is not limited by the usual psychological hindrances pointed out in prior chapters, such as limited storage capacity of attentional memory, limitation of knowledge that is readily available to the conscious introspection at the time of inquiry [16], excessive cognitive load [78], and information overload [14]. The thesis, in this aspect, aligns with the trend of shifting towards automated techniques.

In addition, the thesis extends work on many recent works that begin to utilize natural language processing techniques [12, 64], by applying state-of-the-art NLP techniques to requirements artifacts. My work on eliciting domain knowledge from scenarios and from masked language models employed the state-of-the-art techniques at the time when the work was conducted, and my work on using large language models to support elicitation uses the current state-of-the-art models at the time of writing this thesis.

7.2.2 Emphasizing User-Centered Requirements

There is a growing recognition of the importance of eliciting and satisfying non-functional requirements that reflect user preferences. While existing work in personalization has established the importance of tailoring products to user needs [43, 106], there has been limited research on how to systematically extract and model these preferences as requirements. This thesis bridges this gap by providing specific methods for stakeholder preference extraction and modeling. While requirements research has focused on functional requirements, or broadly on requirements, this thesis emphasizes user-oriented non-functional requirements, particularly stakeholder preferences, to support personalization and preference extraction. Unlike functional requirements of software-based systems, stakeholder preferences describe states of the world that software can observe in order to support user decision making. Thus, advances to elicit and model preferences can lead to a broader impact than conventional requirements engineering research. Moreover, the thesis work aligns with this trend by studying methods for direct elicitation of stakeholder preferences, contrary to many existing approaches that rely on statistical models derived from user-system interaction. The elicitation from scenarios and interviews, in particular, seeks to obtain preferences from users by directly asking the question of "what do users prefer," and incorporates embeddings-assisted automated methods into the elicitation guidance process and analysis stage. For example, users of apartment finding applications may prefer an apartment that is located near a playground, because they have small children, or located near a bus stop, because they require public transportation. While this information may be observable by separate systems, it may not be integrated to support user search and the information can be highly personal and sensitive. I believe the requirements engineering community can benefit from insight into this new world of stakeholder preferences and to discover, encode and predict preference satisfaction across a range of unstudied use cases.

7.2.3 Summary

By positioning the described embeddings-assisted methods introduced in this thesis within the larger requirements engineering picture, the thesis contributes to the ongoing evolution of requirements engineering practices toward more automated, intelligent, and user-centered approaches. The integration of modern techniques into established requirements engineering practices offers a path forward for enabling a software system to capture and satisfy more diverse stakeholder needs in the future.

7.3 Limitations and Future Work

The detailed limitations of each specific work are described in the respective chapters of the work. In this section, we review the limitations of the work as a whole.

7.3.1 Question Generation Integration into Practice

A limitation of the elicitation interview question generation work described in Chapter 6 is the focus on evaluating large language model capacity to be at least as good as human perfromance

with respect to relevancy, clarity, and informativeness. The incorporation of generated questions in a real or simulated interview environment, which may include building the question generation method into an interview chatbot, for example, and recruiting participants to interview and be interviewed, was left unstudied. Hence, we are unaware of the practical challenges that may arise, such as the maximum tolerable latency in question generation, which may disrupt the ability of the interviewer to integrate the generated question into the interview flow, and whether the questions themselves facilitate improvements in requirements yield and quality.

7.3.2 Evaluation of Requirements Completion

Significant work reported in this thesis aims to supplement domain models to produce a more complete version of requirements. However, one problem that is left unstudied is how to evaluate or decide when saturation is reached. Sometimes, developers in practice need to decide whether or not to satisfy all preferences elicited, due to time and resource limitations, and they would need to know when to stop eliciting requirements. For example, in the work of domain knowledge modeling from scenarios and from word embeddings described in Chapter 3 and Chapter 4, we describe methods to increase domain model completeness and to discover additional preferences that other methods may not have been able to elicit, but we did not study when a domain model is considered complete, or when a developer could stop eliciting preferences for the target system. For example, a developer of an apartment finding app may be interested in the noise level as a preference. However, they may not want or need to probe this preference in more detail, for example, by distinguishing the morning noise level or noise produced by a nearby dog barking versus daily traffic. Future work could investigate how developers decide when a model is saturated to understand when elicited requirements have reached a satisfiable level of quantity and specificity.

7.4 Open Questions

In this section, I point out some open questions that may remain within the requirements engineering community that may be interesting future work. The list is not intended to be conclusive in any means, but to shed light on possible future work directions that I myself believe would be of good value.

1. Evaluation Metrics for Automated Requirements Elicitation: There remains a lack of standardized or consistent metrics for evaluating the effectiveness of automated requirements engineering techniques, despite significant work studying automated approaches [201] and using various evaluation criteria, such as completeness and consistency for evaluating specification [104], scalability and satisfaction rate for evaluating prioritization [19], and even machine learning classifiers for evaluating requirements quality [144]. For example, one would ask the question "what qualities a 'good' domain model should satisfy" or "what kind of interviews are considered successful." Moreover, many requirements elicitation practices naturally contain a lot of natural language to be generated, processed and analyzed. While large language models are quite effective at doing this job, its generated contents in the requirements engineering context often needs to be evaluated by human evaluators through

surveys [187], interviews [154], among other techniques [88, 130], and these questions have no "true" or "false" choices, which makes them hard to evaluate with benchmark datasets. Human evaluators can also be biased due to their different views and perspectives, and are susceptible to biased induced by focusing too heavily on coherence, grammar, and certain other aspects, while ignoring other important aspects of the generated contents. On the other hand, evaluating natural language generation contents on generation quality may have established benchmarks [42], but evaluating these contents for more subjective aspects, like novelty, diversity, or adequacy for knowledge advancement, and sometimes their tradeoffs [200], has less established standards. Some recent work has began to address the problem, such as using a five key dimensions evaluation card [136]. Hence establishing standardized and consistent metrics for evaluating automated requirements elicitation is an interesting future direction.

2. Privacy and Ethics:

   Preferences are personal to individual users and cover such topics as an person's income, health status, marital status, and presence/absence of children [178]. Elicitation practices that support the collection of preferences should protect this data as personal information [196]. Privacy-sensitive preferences are generally exclusionary, meaning that stakeholders hold these preferences at the exclusion of holding other preferences, and that stakeholders may be less willing to disclose these sensitive preferences [11]. On the other hand, as preference elicitation becomes more automated and personalized, stakeholder privacy and sensitive preferences may become harder to maintain. For example, models trained on various data sources may inadvertently encode or amplify biases present in their training data. One challenge for RE researchers is to understand how biased models can ignore user preferences, because such models generalize to an artificial norm, which may include opposing views (e.g., people who are unafraid of the police, or people without children). Thus, a challenge is to elicit preferences and use these preferences to evaluate model output when testing applications.

3. Collaboration of Human and AI:

   While much work has sought to develop methods to automate requirements engineering practices via advanced technologies, human also play an essential role in the elicitation practices. Maintaining trustworthy and synergistic collaborations between humans and AI agents is a challenge [117, 118]. For example, this thesis includes work that focus on helping requirements analysts conduct interviews via guided domain elements suggestion and question generation, which provides potential opportunities for human interviewers to collaborate with AI by taking its suggestions during actual interview practices 5, 6. Another piece of work in this thesis describes a pipeline for software developers to utilize automated techniques we outlined to gather stakeholder preferences from scenarios 3. While these work has provided examples for how human and AI may co-exist in requirements practices, the question of how to find the optimal balance between human expertise and AI assistance in requirements engineering remains an open question [1], as well as how to form well-structured definitions and standards for such tasks [141].

7.5 Final Remarks

In summary, my thesis explores how natural language embeddings that encode semantic information can be used to support requirements acquisition by supplementing gaps in stakeholder knowledge and using this knowledge to guide elicitation. Specifically, we explored ways to model domain knowledge from user-authored scenarios and from word embeddings using masked language models, and studied methods to refine interviews as a common requirement artifact by guiding preference elicitation with domain knowledge obtained from word embeddings, and by generating elicitation interview questions with a large language model. There are, of course, a lot left to be studied in this space of work, including open challenges left for future work, such as domain knowledge modeling techniques using more sophisticated language models, the refinement of elicitation interviews with other promising question generation techniques and suggestion methods in real time, or the refinement of requirements artifacts other than interviews with extracted domain knowledge.

Bibliography

[1] K. Ahmad, M. Abdelrazek, C. Arora et. al. "Requirements engineering framework for human-centered artificial intelligence software systems." *Applied Soft Computing 143 (2023): 110455*, 2023. 3

[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. 2.4

[3] S. Apel, D. Batory, C. Kästner, G. Saake. Feature-oriented software product lines. *Springer Publishing Company, Incorporated*, 2013. 2.5

[4] M. Ataei, H. Cheong, D. Grandi, Y. Wang et. al. Elicitron: An LLM Agent-Based Simulation Framework for Design Requirements Elicitation. *arXiv preprint arXiv:2404.16045*, 2024. 2.6, 6.1, 6.2.3

[5] M. Acher, A. Cleve, G. Perrouin et. al., On extracting feature models from product descriptions, *VaMoS '12: Proceedings of the 6th International Workshop on Variability Modeling of Software-Intensive Systems , pages 45-54*, 2012. 2.5

[6] T. Ahmed, P. Devanbu, C. Treude, M. Pradel, "Can LLMs replace manual annotation of software engineering artifacts?". *22nd International Conference on Missing Software Reporistories (MSR25)*, 2025. 6.2.4

[7] T. Alshehri, R. Kirkham, and P. Olivier, "Scenario Co-Creation Cards: A Culturally Sensitive Tool for Eliciting Values," *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20). Association for Computing Machinery, New York, NY, USA, 1–14*, 2020. 6.1

[8] I. Androutsopoulos, P. Malakasiotis. "A survey of paraphrasing and textual entailment methods." *Journal of Artificial Intelligence Research, 38, pp.135-187*, 2010. 6.1

[9] A. Araújo, R. Marcacini. Re-bert: automatic extraction of software requirements from app reviews using bert language model. *In Proceedings of the 36th annual ACM symposium on applied computing, pp. 1321-1327*, 2021. 2.5

[10] R. Artstein, M. Poesio. Inter-coder agreement for computational linguistics, *Computational Linguistics*, 34(4): 555-596, 2008. 3.2.2

[11] P. Anthonysamy, A. Rashid, and R. Chitchyan. "Privacy requirements: present & future." *2017 IEEE/ACM 39th international conference on software engineering: software engineer-*

ing in society track (ICSE-SEIS). IEEE, 2017. 2

[12] 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. 2.4, 2.5, 7.2.1

[13] 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.

[14] M. Aljukhadar, S. Senecal, C. Daoust. Using recommendation agents to cope with information overload. *International Journal of Electronic Commerce 17, no. 2: 41-70*, 2012. 2.6, 7.2.1

[15] C. Arora, M. Sabetzadeh, S. Nejati, L. Briand. An Active Learning Approach for Improving the Accuracy of Automated Domain Model Extraction, ACM Transactions on Software Engineering Methodology, 28(1): Article 4, 2019. 2.4

[16] R. Agarwal, M. R. Tanniru, "Knowledge Acquisition Using Structured Interviewing: An Empirical Investigation," *Journal of Management Information Systems 7 (1): 123–40. doi:10.1080/07421222.1990.11517884*, 1990. 2.6, 6.2.3, 7.2.1

[17] K.Barton, Elicitation Techniques: Getting People to Talk About Ideas They Don't usually talk about, *Theory and Research in Social Education, 43(2), 179–205*, 2015. 2.6, 5.1

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

[19] F. Bukhsh, Z. Bukhsh, and M. Daneva. "A systematic literature review on requirement prioritization techniques and their empirical evaluation." *Computer Standards Interfaces 69: 103389*, 2020. 1

[20] M. Bragilovski, F. Dalpiaz, and A. Sturm. From user stories to domain models: Recommending relationships between entities. In *CEUR Workshop Proceedings, vol. 3378. CEUR-WS*, 2023. 5.1

[21] D. Bajaj, A. Goel, S. Gupta, and H. Batra. "MUCE: a multilingual use case model extractor using GPT-3." *International Journal of Information Technology 14, no. 3: 1543-1554*, 2022. 2.5

[22] A. Ravichander, E. Hovy, K. Suleman, A. Trischler, J. C. K. Cheung. "On the Systematicity of Probing Contextualized Word Representations: The Case of Hypernymy in BERT." 9th Joint Conference on Lexical and Computational Semantics, pages 88–102, 2020. 2.5

[23] I. Beltagy, K. Lo, A. Cohan, "SciBERT: A Pretrained Language Model for Scientific Text." *EMNLP*, 2019. 4.4.4

[24] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901. 2.5, 6.4.1

[25] D.D. Brewer. Supplementary interviewing techniques to maximize output in free listing tasks.

Field methods, 14(1): 108-118, 2002.

[26] M. Broy, "Domain Modeling and Domain Engineering: Key Tasks in Requirements Engineering." *Perspectives on the Future of Software Engineering*, 2013. 2.5, 5.1

[27] B. Boehm, R. Turner. Balancing Agility and Discipline: A Guide for the Perplexed. *Addison-Wesley*, 2003. 2.2

[28] R. A. Bradley and M. E. Terry. "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons." *Biometrika 39, no. 3/4 (1952): 324–45.*, 1952. 6.2.5

[29] S. Brandon, S. Wells, C.Seale. Science-based interviewing: Information elicitation. *Journal of Investigative Psychology and Offender Profiling 15, no. 2: 133-148*, 2018. 2.6, 5.1

[30] M. Bano, D. Zowghi, A. Ferrari, P. Spoletini, and B. Donati, "Learning from mistakes: An empirical study of elicitation interviews performed by novices," *2018 IEEE 26th International Requirements Engineering Conference (RE). IEEE, 2018, pp. 182–193*, 2018. 2.6, 6.2.3

[31] M. Bano, D. Zowghi, A. Ferrari, P. Spoletini, and B. Donati. Teaching requirements elicitation interviews: an empirical study of learning from mistakes. *Requirements Engineering 24: 259-289*, 2019. 2.6, 5.1

[32] 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. 2.5

[33] Y., Chenyang, R. Brower-Sinning, G. Lewis, C. Kästner, and T. Wu. Capabilities for better ml engineering. *arXiv preprint arXiv:2211.06409*, 2022. 2.4

[34] A. Colson, R. Cooke. Expert elicitation: using the classical model to validate experts' judgments. *Review of Environmental Economics and Policy*, 2018. 5.1

[35] B. Chen, K. Chen, S. Hassani, Y. Yang et. al. "On the use of GPT-4 for creating goal models: an exploratory study." In *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW), pp. 262-271. IEEE*, 2023. 2.5

[36] D. Carrizo, O. Dieste, and N. Juristo, "Systematizing requirements elicitation technique selection," *Inf. Softw. Technol. 56, 6 (June, 2014), 644–669*, 2014. 6.1

[37] K. Charmaz, Constructing Grounded Theory, 2nd ed. SAGE, 2014. 3.2.2

[38] J. T. Cacioppo, W. von Hippel, J. M. Ernst, "Mapping cognitive structures and processes through verbal content: The thought-listing technique," *Journal of Consulting and Clinical Psychology, 65(6), 928–940.* , 1997.

[39] M. Christel, K. Kang. Issues in requirements elicitation. *DTIC Document*, 1992. 5.1

[40] J. Chiu, E. Nichols, Named Entity Recognition with Bidirectional LSTM-CNNs, *Transactions of the Association for Computational Linguistics*, 4:357-370, 2016. 2.5

[41] C. Cortes, V. Vapnik, Support-vector networks. *Machine learning, 20(3), 273-297*, 1995. 2.5

[42] W. Chiang, L. Zheng, Y. Sheng et. al., "Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference", *Forty-first International Conference on Machine Learning*. 2024. 1

[43] S. M. Davis. *Future Perfect*. Adison-Wesley, Reading, MA, 1987. 2.2, 7.2.2

[44] J. Devlin, M. Chang, K. Lee, K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv:1810.04805*, 2018. 2.5, 3.2.4

[45] 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. 2.4, 3.2.4

[46] A. Davis, O. Dieste, A. Hickey, N. Juristo and A. M. Moreno, Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review, *14th IEEE International Requirements Engineering Conference (RE'06), Minneapolis/St. Paul, MN, USA, pp. 179-188*, 2006. 5.1

[47] B. Douglas, P.J. Ewell, and M. Brauer. "Data quality in online human-subjects research: Comparisons between MTurk, Prolific, CloudResearch, Qualtrics, and SONA." *PLoS ONE 18(3): e0279720*, 2023. 6.5

[48] B. Donati, A. Ferrari, P. Spoletini, and S. Gnesi, "Common mistakes of student analysts in requirements elicitation interviews," *International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2017, pp. 148–164*, 2017. 2.6, 6.2.3

[49] C. Detweiler, M. Harbers, Value Stories: Putting Human Values into Requirements Engineering, *REFSQ Workshops*, 2-11, 2014. 2.4, 7.2.1

[50] A. Dardenne, A. van Lamsweerde, S. Fickas. Goal-directed requirements acquisition, Science of computer programming, 20(1-2): 3-50, 1993. 2.1

[51] I. Dagan, D. Roth, F. Zanzotto and M. Sammons. "Recognizing textual entailment: Models and applications", *Springer Nature*, 2022. 6.1

[52] H. Dai, Y. Song, H. Wang. Ultra-Fine Entity Typing with Weak Supervision from a Masked Language Model, 59th Anual Meeting of the Association for Computational Linguistics, pp. 1790–1799. 2021. 2.5

[53] C. Duhigg, Smarter faster better: The transformative power of real productivity, *Random House Publishing, New York, NY*, 2016. 2.6, 5.1

[54] A. Ettinger. "What BERT Is Not: Lessons from a New Suite of Psycholinguistic Diagnostics for Language Models." *Transactions of the Association for Computational Linguistics*, 8:34–48, 2020. 4.1

[55] A. Ferrari, F. Dell'Orletta, G.Spagnolo, and S. Gnesi. Measuring and improving the completeness of natural language requirements. In Camille Salinesi and Inge van de Weerd, editors, *REFSQ*, 2014. 5.1

[56] C. Fellbaum. "WordNet and wordnets." In: Brown, Keith et al. (eds.), Encyclopedia of Language and Linguistics, 2nd ed., Oxford: Elsevier, 665-670, 2005.

[57] 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. 3.2.4

[58] M. Feffer, H. Heidari, and Z. C Lipton. “Moral machine or tyranny of the majority?” *arXiv preprint arXiv:2305.17319*, 2023. 6

[59] 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. 1, 2.2, 7.2.1

[60] 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. 2.4

[61] A. Ferrari, P. Spoletini, M. Bano and D. Zowghi, “Learning Requirements Elicitation Interviews with Role-Playing, Self-Assessment and Peer-Review,” *2019 IEEE 27th International Requirements Engineering Conference (RE), Jeju, Korea (South)*, 2019. 5.2.3, 5.5.2, 6.2.3, 6.5

[62] A. Ferrari, P. Spoletini, and S. Debnath. How do requirements evolve during elicitation? An empirical study combining interviews and app store analysis. *Requirements Engineering 27, no. 4: 489-519*, 2022. 2.6, 5.1

[63] A. Ferrari, P. Spoletini, and S. Gnesi. Ambiguity and tacit knowledge in requirements elicitation interviews. *Requirements Engineering 21, no. 3: 333-355*, 2016. 2.6, 5.1, 7.2.1

[64] B. Görer and F. B. Aydemir, “Generating Requirements Elicitation Interview Scripts with Large Language Models,” *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW), Hannover, Germany, 2023, pp. 44-51, doi: 10.1109/REW57809.2023.00015.*, 2023. 2.6, 6.1, 6.2.3, 7.2.1

[65] B. Görer and F. B. Aydemir, “GPT-Powered Elicitation Interview Script Generator for Requirements Engineering Training”, *arXiv:2406.11439*, 2024. 2.6, 6.2.3

[66] G. Gazdar. “Phrase Structure Grammar” In: Jacobson, P., Pullum, G.K. (eds) *The Nature of Syntactic Representation.* Synthese Language Library, vol 15. Springer, 1982. 2.4

[67] T. Gemkow, M. Conzelmann, K. Hartig, A. Volesang, Automatically glossary term extraction form large-scale requirements specifications. *26th IEEE RE Conference*, 2018. 2.3

[68] M. Garima, M. Cevik, Y. Khedr, D. Parikh, and A. Basar, Named Entity Recognition on Software Requirements Specification Documents. *Canadian Conference on AI*, 2021. 2.5

[69] Y. Ge, W. Hua, K. Mei, J. Tan, S. Xu, Z. Li, Y. Zhang. “Openagi: When llm meets domain experts”. *Advances in Neural Information Processing Systems*, 2023. 6.4.4

[70] C. Gallego, C. Jaramillo, QUARE: towards a question-answering model for requirements elicitation. *Autom Softw Eng 30, 25*, 2023. 5.1

[71] V. Gupta, P. Kenny, P. Ouellet and T. Stafylakis, I-vector-based speaker adaptation of deep neural networks for french broadcast audio transcription. *In 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP) (pp. 6334-6338). IEEE*, 2014. 2.6

[72] M. Glinz, On Non-Functional Requirements, In Proc. *15th International Conference on Requirements Engineering*, pp. 21-26, 2007. 1, 2.1

[73] 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.

[74] 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. 7.1.1

[75] C. Guo, P. Pleiss, Y. Sun, K. Weinberger, On calibration of modern neural networks. *International conference on machine learning, PMLR*, 2017.

[76] E. Groen; N. Seyff; R. Ali, F. Dalpiaz et. al., The Crowd in Requirements Engineering: The Landscape and Challenges, *IEEE Software, Volume: 34, Issue: 2, Mar.-Apr. 2017*, 2017. 2.3

[77] J. Horkoff, F.B. Aydemir, E. Cardoso, et al. Goal-oriented requirements engineering: an extended systematic mapping study, *Requirements Eng 24, 133–160*, 2019. 2.4

[78] P. Hanway, The effects of cognitive load for investigative interviewers. *PhD diss., University of Portsmouth*, 2020. 2.6, 7.2.1

[79] A. Hickey, A. Davis. Elicitation technique selection: how do experts do it?. In *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003., pp. 169-178. IEEE*, 2003. 2.6

[80] T. Huang, V. Kaulagi, M. B. Hosseini and T. Breaux, "Mobile Application Privacy Risk Assessments from User-authored Scenarios," *2023 IEEE 31st International Requirements Engineering Conference (RE), Hannover, Germany, 2023, pp. 17-28, doi: 10.1109/RE57278.2023.00012*, 2023. 5.5.1

[81] 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. 2.5

[82] I. Hadar, P. Soffer, and K. Kenzi. The role of domain knowledge in requirements elicitation via interviews: an exploratory study. *Requirements Engineering 19: 143-159*, 2014. 2.6

[83] Z. Huang, W. Xu, K. Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR, abs/1508.01991*, 2015. 2.5

[84] X. Han, M. Zhou, M. J. Turner, and T. Yeh, "Designing Effective Interview Chatbots: Automatic Chatbot Profiling and Design Suggestion Generation for Chatbot Debugging," *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 389, 1–15*, 2021. 2.6, 6.1, 6.2.3

[85] M. Ibrahim and R. Ahmad, Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques, *Second International Conference on Computer Research and Development, pp. 200-204, doi: 10.1109/ICCRD.2010.71*, 2010.

[86] M. Jackson, The World and the Machine. In Proc. *17th International Conference on Software Engineering*, pp. 283–292, 1995. 2.1

[87] R. Jiang, R.E. Banchs, H. Li, Evaluating and Combining Named Entity Recognition Systems,

Proceedings of the Sixth Named Entity Workshop, 2016.

[88] L. Jiang, A. Eberlein, B. Far, M. Mousavi. "A methodology for the selection of requirements engineering techniques". *Software Systems Modeling, 7(3), 303-328*, 2008. 1

[89] D. Jurafksy and J. Martin. Speech and Language Processing, 2nd ed. Prentice Hall, 2008. 3.2.5

[90] N. Jha, A. Mahmoud. "Mining non-functional requirements from App store reviews," *Empirical Software Engineering*, 24: 3659–3695, 2019. 2.5

[91] I. Jureta, J. Mylopoulos and S. Faulkner, "Revisiting the Core Ontology and Problem in Requirements Engineering," *2008 16th IEEE International Requirements Engineering Conference, Barcelona, Spain, 2008, pp. 71-80, doi: 10.1109/RE.2008.13*, 2008. 2.1

[92] J. Johnson, In-Depth Interviewing, in *Handbook of Interview Research: Context and Method, Sage, pp. 103-119*, 2022.

[93] 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. 2.5

[94] J. Johnson, S. Weller, Elicitation Techniques for Interviewing, In *Handbook of Interview Research, 491-514, SAGE Publications, Inc., ISBN: 9780761919513*, 2011.

[95] M. Kuller, P. Beutler, J. Lienert, Preference change in stakeholder group-decision processes in the public sector: Extent, causes and implications, *European Journal of Operational Research, Volume 308, Issue 3, Pages 1268-1285, ISSN 0377-2217*, 2023.

[96] V. Keswani, V. Conitzer, W. Sinnott-Armstrong et. al. "Can AI Model the Complexities of Human Moral Decision-making? A Qualitative Study of Kidney Allocation Decisions". *In Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (pp. 1-17)*, 2025. 6

[97] T. Kojima, S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems 35: 22199-22213*, 2022. 5.4.3

[98] 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. 2.4

[99] 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. 2.2

[100] J. Khan, L. Liu, L. Wen, R. Ali, Crowd Intelligence in Requirements Engineering: Current Status and Future Directions, *Knauss, E., Goedicke, M. (eds) Requirements Engineering: Foundation for Software Quality. REFSQ 2019*, 2019. 2.3

[101] Y. Kosyakova, L. Olbrich, J.W. Sakshaug, S. Schwanhäuser, "Positive Learning or Deviant Interviewing? Mechanisms of Experience on Interviewer Behavior, Journal of Survey Statistics and Methodology, Volume 10, Issue 2, April 2022, Pages 249–275, https://doi.org/10.1093/jssam/smab003", 2022. 5.5.2

[102] H. Kallio, A.M. Pietilä, M. Johnson, and M. Kangasniemi, "Systematic methodological review: developing a framework for a qualitative semi-structured interview guide". *Journal of advanced nursing, 72(12), pp.2954-2965*, 2016. 2.6

[103] N. Kassner and H. Schütze. "Negated and Misprimed Probes for Pretrained Language Models: Birds Can Talk, But Cannot Fly." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7811–7818, Online. Association for Computational Linguistics*, 2020. 6.4.1

[104] A. Khwaja and Jo. Urban. "A synthesis of evaluation criteria for software specifications and specification techniques." *International Journal of Software Engineering and Knowledge Engineering 12.05: 581-599*, 2002. 1

[105] S. Kühne, S, "Interpersonal Perceptions and Interviewer Effects in Face-to-Face Surveys". *Sociological Methods & Research, 52(1), 299-334. https://doi.org/10.1177/0049124120926215*, 2023. 5.5.2

[106] A. Kumar. From mass customization to mass personalization: a strategic transformation. International *Journal Flexible Manufacturing Systems*, 19:533-547, 2007. 2.2, 7.2.2

[107] A. van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software,* Wiley & Sons, 2009. 4.1

[108] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, C. Dyer. Neural Architectures for Named Entity Recognition, NAACL, 2016. 2.5

[109] S. Lim, A. Henriksson, J. Zdravkovic, Data-Driven Requirements Elicitation: A Systematic Literature Review, *SN Computer Science. 2, 16*, (2021) 2.4

[110] 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. 2.4

[111] J.R. Landis, G.G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159—174, 1977. 3.2.2

[112] 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. 2.1, 2.4

[113] S. Liaskos, S. A. McIlraith, S. Sohrabi, J. Mylopoulos, Representing and reasoning about preferences in requirements engineering, *Requir. Eng. 16*, 3: 227-249, 2011. 2.1, 2.4

[114] J. Lafferty, A. McCallum, F.C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data, 2001. 2.5

[115] M. Linares-Vásquez, C. McMillan, D. Poshyvanyk et al. On using machine learning to automatically classify software applications into domain categories. *Empir Software Eng 19, 582–618*, 2014. 2.4

[116] S. Liaskos, S.A. McIlraith, S. Sohrabi et al. "Representing and reasoning about preferences in requirements engineering". *Requirements Eng 16, 227–249 (2011). https://doi.org/10.1007/s00766-011-0129-9*, 2011. 2.1, 2.4

[117] D. Lo, "Trustworthy and synergistic artificial intelligence for software engineering: Vision and roadmaps," *in IEEE/ACM International Conference on Software Engineering: Future of Software Engineering, ICSEFoSE 2023, Melbourne, Australia, May 14-20, 2023. IEEE, 2023, pp. 69–85*, 2023. 3

[118] D. Lo, "Requirements Engineering for Trustworthy Human-AI Synergy in Software Engineering 2.0." *2024 IEEE 32nd International Requirements Engineering Conference (RE). IEEE*, 2024. 3

[119] Yinhan Liu, Myle Ott, Naman Goyal, et. al. RoBERTa: A Robustly Optimized BERT Pretraining Approach, *10.48550/ARXIV.1907.11692*, 2019. 4.3, 5.2.1

[120] P. Lewis, E. Perez, A. Piktus et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". *Advances in neural information processing systems. 2020;33:9459-74*, 2020. 6.6

[121] L. Litman, J. Robinson, "Conducting Online Research on Amazon Mechanical Turk and Beyond," *SAGE*, 2020. 3.5

[122] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys 55, no. 9: 1-35*, 2023. 5.4.3

[123] I. Lee, W. Yoon, S. Kim, D. Kim, Sunkyu Kim, Chan. So, J. Kang, BioBERT: a pre-trained biomedical language representation model for biomedical text mining, *Bioinformatics, Volume 36, Issue 4, 15 February 2020, Pages 1234–1240*, 2020. 4.4.4

[124] 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. 2.3, 2.4

[125] P. K. Murukannaiah, N. Ajmeri and M. P. Singh, "Toward Automating Crowd RE," *2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 2017, pp. 512-515, doi: 10.1109/RE.2017.74*, 2017. 2.3

[126] M. McHugh. "Interrater reliability: the kappa statistic". *Biochem Med (Zagreb). 2012;22(3):276-82. PMID: 23092060; PMCID: PMC3900052*, 2012. 6.3.2

[127] W. Maalej, H. Happel, A. Rashid, "When Users Become Collaborators: Towards Continuous and Context-Aware User Input", *Proc. 24th ACM SIGPLAN Conf. Object-Oriented Programming Systems Languages and Applications (OOPSLA 09), pp. 981-990*, 2009. 2.3, 7.2.1

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

[129] 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. 2.4, 3.2.4

[130] D. Mishra, A. Mishra, and A. Yazici. "Successful requirement elicitation by combining requirement engineering techniques." *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT). IEEE*, 2008. 1

[131] 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. 2.3, 7.1.1

[132] I. Morales-Ramirez, A. Perini and M. Ceccato, "Towards Supporting the Analysis of Online Discussions in OSS Communities: A Speech-Act Based Approach" *Information Systems Engineering in Complex Environments, Springer, pp. 215-232*, 2014. 2.3

[133] 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. 2.1

[134] S. Mishra and A. Sharma, On the Use of Word Embeddings for Identifying Domain Specific Ambiguities in Requirements, *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW), pp. 234-240, doi: 10.1109/REW.2019.00048*, 2019. 4.1

[135] J. Mylopoulos, L. Chung, B. Nixon, "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. IEEE Trans. Soft. Eng., 18(6): 483-497, 1992. 2.1

[136] Q. Ma, D. Zhao, Z. Zhao, et. al. "SPHERE: An Evaluation Card for Human-AI Systems." *arXiv preprint arXiv:2504.07971*, 2025. 1

[137] S. Nifterik, Exploring the Potential of Large Language Models in Supporting Domain Model Derivation from Requirements Elicitation Conversations, 2024. 2.4

[138] 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. 2.4

[139] D. Nadeau, P. Turney, S. Matwin, Unsupervised Named Entity Recognition: Generating Gazetteers and Resolving Ambiguity. In *Proc. Canadian Conference on Artificial Intelligence*, 2006. 2.5

[140] P. N. Otto and A. I. Anton, Addressing Legal Requirements in Requirements Engineering, *15th IEEE International Requirements Engineering Conference (RE 2007), pp. 5-14, doi: 10.1109/RE.2007.65*, 2007. 4.1

[141] L. Odong, A. Perini, and A. Susi. "Requirements engineering for collaborative artificial intelligence systems: a literature survey." *International Conference on Research Challenges in Information Science. Cham: Springer International Publishing*, 2022. 3

[142] OpenTable, Inc. "Global Fast Facts," Q4 2019. https://press.opentable.com/static-files/3f990a9c-0702-4496-9b80-5b90198d056a 1

[143] L. Ouyang, J. Wu, X. Jiang et al. "Training language models to follow instructions with human feedback". *Advances in neural information processing systems, 2022 Dec 6;35:27730-44*, 2022. 6.6

[144] E. Parra, C. Dimou, J. Llorens, V. Moreno, A. Fraga, "A methodology for the classification of quality of requirements using machine learning techniques." *Information and Software Technology, 67, 180-195*, 2015. 1

[145] C. Palomares, Z. Franch, C. Quer et al. The state-of-practice in requirements elicitation: an extended interview study at 12 companies. *Requirements Eng 26, 273–299*, 2021. 2.6

[146] C. Pacheco, I. Garcia, M. Reyes. Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques, *IET Software*, 2018. 2.4

[147] N. Patil, A. Patil, B.V. Pawar, Named Entity Recognition using Conditional Random Fields, *Procedia Computer Science, Volume 167*, 2020. 3.2.4

[148] D. Peppers, M. Rogers. The One to One Future: Building Relationships One Customer at a Time. Currency Doubleday. 1993. 2.2

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

[150] C. Potts, K. Takahashi, and A. Anton. "Inquiry-based requirements analysis." *IEEE software 11.2 (1994): 21-32*, 1994. 2.6, 6.2.3

[151] 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. 2.4

[152] 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. 3.2.4, 3.2.4

[153] K. Ronanki, C. Berger, J. Horkoff. Investigating ChatGPT's Potential to Assist in Requirements Elicitation Processes. In *49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 354-361. IEEE*, 2023. 5.1

[154] J. Reichental. "An evaluation of the effectiveness of interview techniques in the elicitation of tacit knowledge for requirements engineering in small software projects." *Nova Southeastern University*, 2006. 1

[155] 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. 2.4

[156] S. Ruder, M. Peters, S. Swayamdipta, T. Wolf, Transfer Learning in Natural Language Processing, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, 2019. 4.1

[157] J. Saldaña. The Coding Manual for Qualitative Researchers, *SAGE Publications*, 2012. 3.2.2, 3.2.3, 6.2.2

[158] T. Spijkman, X. Bondt, F. Dalpiaz, and S. Brinkkemper. Summarization of Elicitation Conversations to Locate Requirements-Relevant Information. In *International Working Conference on Requirements Engineering: Foundation for Software Quality, pp. 122-139. Cham: Springer Nature Switzerland*, 2023. 5.1

[159] N. Seyff, S. Betz, I. Groher et. al. Crowd-focused semi-automated requirements engineering for evolution towards sustainability. *Proceedings of 26th RE@Next Conference*, 2018. 2.3

[160] H. Schmid, "Part-of-speech tagging with neural networks." *arXiv preprint cmp-lg/9410018*, 1994. 2.6

[161] A. Sleimi, M. Ceci, N. Sannier, M. Sabetzadeh, L. C. Briand, J. Dann. A Query System for Extracting Requirements-related Information from Legal Texts, IEEE RE, 2019. 2.5

[162] M. Sclar, Y. Choi, Y. Tsvetkov, and A. Suhr. "Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting." In *arXiv preprint arXiv:2310.11324*, 2023. 6.4.1

[163] S. Victor, L. Debut, J. Chaumond, T. Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." *ArXiv abs/1910.01108*, 2019. 3.2.4

[164] P. Spoletini, A. Ferrari, M. Bano, D. Zowghi, and S. Gnesi. Interview review: An empirical study on detecting ambiguities in requirements elicitation interviews. In *Requirements Engineering: Foundation for Software Quality: 24th International Working Conference, REFSQ 2018, Utrecht, The Netherlands, March 19-22, 2018, Proceedings 24, pp. 101-118. Springer International Publishing*, 2018. 2.6

[165] C. Surana, D. Gupta, S. Shankar. Intelligent chatbot for requirements elicitation and classification. In *4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), pp. 866-870. IEEE*, 2019. 5.1

[166] 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. 2.4

[167] F. Sarro, M. Harmna, Y. Jia, Y. Zhang, Customer rating reactions can be predicted purely using app features. *Proceedings of 26 IEEE RE Conference*, 2018. 2.3

[168] S. Singh, K. Jiang, K. Bhasin et. al., RACER: An LLM-powered Methodology for Scalable Analysis of Semi-structured Mental Health Interviews, *arXiv:2402.02656*, 2024. 2.6, 6.1, 6.2.3

[169] 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. 2.2

[170] A. Sutcliffe and N. Maiden, The domain theory for requirements engineering, *in IEEE Transactions on Software Engineering, vol. 24, no. 3, pp. 174-196, March 1998, doi: 10.1109/32.667878*, 1998. 2.4, 4.1, 7.2.1

[171] R. Saini, G. Mussbacher, J.L.C. Guo, J. Kienzle. Towards Queryable and Traceable Domain Models. *IEEE 28th International Requirments Engineering Conference*, 2018. 2.4

[172] A. Sree-Kumar, E. Planas, and R. Clarisó. Extracting software product line feature models from natural language specifications. *In Proceedings of the 22nd International Systems and Software Product Line Conference - Volume 1 (SPLC '18). Association for Computing Machinery, New York, NY, USA, 43–53. https://doi.org/10.1145/3233027.3233029*, 2018. 2.5

[173] A. Sutcliffe and P. Sawyer, Requirements elicitation: Towards the unknown unknowns, *2013 21st IEEE International Requirements Engineering Conference (RE), pp. 92-104, doi: 10.1109/RE.2013.6636709*, 2013. 4.1

[174] D. Stewart, P. Shamdasani. Focus groups: Theory and practice, *Vol. 20. Sage publications*, 2014. 2.6

[175] 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. 2.4

[176] V. Sugumaran, M. Tanniru, and V. Storey. Identifying software components from process requirements using domain model and object libraries, 1999. 2.4, 4.1, 7.2.1

[177] 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. 2.4

[178] M. Teltzrow and A. Kobsa. "Impacts of user privacy preferences on personalized systems: a comparative study." *Designing personalized user experiences in eCommerce. Dordrecht: Springer Netherlands, 2004. 315-332*, 2004. 2

[179] A. Tversky, D. Kahneman. Judgment under Uncertainty: Heuristics and Biases. *Science. 185 (4157): 1124–1131*, 1974. 4.4.1

[180] W. Tai, H. Kung, X. Dong, et al. exBERT: Extending pre-trained models with domain-specific vocabulary under constrained training resources. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020. 4.1, 4.4.4, 5.4.3

[181] 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. 2.5

[182] 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. 2.5

[183] A. van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley, 2009. 1, 2.1

[184] K. Valmeekam, A. Olmo, S. Sreedharan, S. Kambhampati, "Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change)", *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. 6.4.4

[185] 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. 2.5, 2.6

[186] 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. 3.2.4

[187] S. Wagner, D. Fernández, M. Felderer, et. al. "Status quo in requirements engineering: A theory and a global family of surveys". *ACM Transactions on Software Engineering and Methodology (TOSEM), 28(2), 1-48*, 2019. 1

[188] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, D. C. Schmidt, "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT", *arXiv preprint arXiv:2302.11382*, 2023. 6.4.1

[189] C. Wang, X. He. "BiRRE: Learning Bidirectional Residual Relation Embeddings for Su-

pervised Hypernymy Detection." 58th Annual Meeting of the Association for Computational Linguistics, pages 3630–3640, 2020. 2.5

[190] K. Weiss, T.M. Khoshgoftaar, D. Wang, A survey of transfer learning. *J Big Data 3, 9 (2016). https://doi.org/10.1186/s40537-016-0043-6*, 2016. 4.1

[191] K. Wongsuphasawat, Y. Liu, and J. Heer. "Goals, process, and challenges of exploratory data analysis: An interview study." *arXiv preprint arXiv:1911.00568*, 2019. 2.6

[192] G. Williams, A. Mahmoud, Modeling user concerns in the app store: a case study on the rise and fall of Yik Yak. *Proceedings of 26 IEEE RE Conference*, 2018. 2.3

[193] Y. Xie, K. Aggarwal, A. Ahmad. "Efficient continual pre-training for building domain specific large language models". In *Findings of the Association for Computational Linguistics ACL 2024 2024 Aug (pp. 10184-10201)*, 2024. 6.6

[194] T. Yoshioka, I. Abramovski, C. Aksoylar, Z. Chen et. al., Advances in online audio-visual meeting transcription. *In 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU) (pp. 276-283). IEEE*, 2019. 2.6

[195] 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. 2.5, 3.2.4

[196] E. Yu and L. Cysneiros. "Designing for privacy and other competing requirements." *2nd Symposium on Requirements Engineering for Information Security (SREIS'02), Raleigh, North Carolina*, 2002. 2

[197] R. K. Yin, Case study research: Design and methods, vol. 5. Sage, 2009. 3.5, 5.5.1, 5.5.2, 5.5.3, 6.5

[198] J. Youn, T. Naravane, I. Tagkopoulos. "Using Word Embeddings to Learn a Better Food Ontology." Frontiers in Artificial Intelligence, 2020. 2.5

[199] C. Yang, R. Rustogi, R. Brower-Sinning, G. Lewis, C. Kaestner, and T. Wu. Beyond Testers' Biases: Guiding Model Testing with Knowledge Bases using LLMs. *In Findings of the Association for Computational Linguistics: EMNLP 2023, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 13504–13519. https://doi.org/10.18653/v1/2023.findings-emnlp.901*, 2023. 2.4

[200] C. Yang, T. Xiao, M. Shavlovsky, et. al. "Orbit: A Framework for Designing and Evaluating Multi-objective Rankers." *In Proceedings of the 30th International Conference on Intelligent User Interfaces (pp. 1093-1106)*, 2025. 1

[201] L. Zhao, W. Alhoshan, A. Ferrari, et. al. "Natural language processing for requirements engineering: A systematic mapping study". *ACM Computing Surveys (CSUR), 54(3), 1-41*, 2021. 1

[202] D. Zowghi, C. Coulin, Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In *Engineering and Managing Software Requirements, Chapter 2, Springer, Berlin, Heidelberg*, 2005. 1, 2.6, 5.1, 6.1

[203] L. Zheng, W. Chiang, Y. Sheng et al. "Judging llm-as-a-judge with mt-bench and chatbot

arena". *Advances in Neural Information Processing Systems. 2023 Dec 15;36:46595-623*, 2023. 6.6

[204] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning, pp. 12697-12706. PMLR*, 2021. 5.4.3

[205] Data and tools for Chapter 3: `https://figshare.com/articles/software/Stakeholder_Preference_Extraction_from_Scenarios_-_Code_and_Datasets/24431527` 3.6, 7.1.1

[206] Data and tools for Chapter 4: `https://drive.google.com/drive/folders/1o7hG0N63bl5Gpuuvu5GsTFws4Hb4d4NL?usp=sharing` 4.3, 4.5, 7.1.1

[207] Data and tools for Chapter 5: https://doi.org/10.6084/m9.figshare.25464373.v1. 1

[208] Data and tools for Chapter 6: https://figshare.com/account/items/29206232.