Domain Model Extraction from User-authored Scenarios and Word Embeddings

Yuchen Shen, Travis Breaux
Institute for Software Research
Carnegie Mellon University
Pittsburgh, United States
yuchenshen@cmu.edu, breaux@cs.cmu.edu

***Abstract*—Domain models are used by requirements analysts to rationalize domain phenomena into discrete entities that drive requirements elicitation and analysis. Domain models include entities, actors or agents, their actions, and desired qualities assigned to states in the domain. Domain models are acquired through a wide range of sources, including interviews with subject matter experts, and by analyzing text-based scenarios, regulations and policies. Requirements automation to assist with elicitation or text analysis can be supported using masked language models (MLM), which have been used to learn contextual information from natural language sentences and transfer this learning to natural language processing (NLP) tasks. The MLM can be used to predict the most likely missing word in a sentence, and thus be used to explore domain concepts encoded in a word embedding. In this paper, we explore an approach of extracting domain knowledge from user-authored scenarios using typed dependency parsing techniques. We also explore the efficacy of a complementary approach of using a BERT-based MLM to identify entities and associated qualities to build a domain model from a single-word seed term.**

***Index Terms*—requirements, domain model, word embedding**

I. Introduction

In requirements engineering, the requirements analyst seeks to understand the problem world, or domain, to best identify requirements for the solution [16]. 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 [26]. With regard to textual representations of domain knowledge, such as scenarios, the text may be incomplete or ambiguous [21], [22]. 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), however, new opportunities exist to automate elicitation and domain analysis tasks.

Neural models in particular have shown strong performance across many natural language processing (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 [36]. This is particularly true of word embeddings, which are often constructed using unsupervised machine learning methods over large corpora [21], [25], [32].

In this paper, we explore two approaches to acquire domain models: (1) we collect a user-authored scenario corpus in the directory service domain and analyze it to extract domain knowledge using typed dependency parsing, and (2) 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 [8], 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 each approach and how they can be used complementary to each other.

The paper is organized as follows: we review related work in Section II and background in Section III, before presenting our approach in Section IV. Next, we present results in Section V with discussion in Section VI and the conclusion and future work in Section VII.

II. Related Work

We now discuss related work on domain model development, including work in typed dependencies, phrase structure grammars and named entity recognition.

Domain models, which describe knowledge about specific application domains [6], are built to capture requirements for systems. Techniques exist for automatically extracting domain model elements, such as using rules based on information retrieval, using natural language dependency parsing [2], 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 [20]. Dependencies have been used in requirements engineering to extract assertions from NL specifications for use in formal verification [28], legal meta-data from regulations [27],

This research was funded in part by NSF Award #2007298.

and software features from user manuals [23]. They have also been used to classify requirements as either functional or non-functional [7], [15], and to demarcate requirements in a specification [1]. Conceptual models have been extracted from user stories using syntactic rules that resemble typed dependencies (e.g., nsubj and nmod) [24]. 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 applications of typed dependencies require an analysis of a large number of examples, or a narrow domain with few syntactic variations to express requirements-related information of interest. While current work using typed dependencies with machine learning classification shows promise [1], [7], [15], more work is needed to extract requirements-related entities from text, including approaches based on custom models.

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 [12]. 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 [2]. This approach has been extended using active learning to reduce false positives by 45% to yield a .96 precision [3]. Saini et al. combine named entity recognition and co-reference resolution with rules to extract classes, relationships, attributes and cardinalities from problem descriptions [30].

Sleimi et al. describe a tool to automatically extract requirements information from legal texts using a concept typology [29]. 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.

In natural language processing, named entity recognition is the identification of entities in text that conform to a small number of predefined types, e.g., person, organization, date, etc. [17]. 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 [10].

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 [37]. The extension is build using a mapping function that maps word embedding vectors to a target class, and performance is evaluated using measures of granularity and cohesiveness. Ravichander et al., inspired by the Cloze task, describe using MLMs to discover hypernyms, which are more general categories for words [34]. Wang and He identify hypernyms using bidirectional residual relation embeddings (BiRRE) to create a latent projection model with negative regularization [35]. 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.

III. Background

We now discuss background on Masked Language Models, and Bidirectional Encoder Representations from Transformers.

A. Masked Language Model

The Masked Language Model (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 [9]. 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.

B. Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based language representation model that utilizes the attention mechanism to learn word embeddings. BERT is pre-trained with two unsupervised tasks, namely a Masked Language Model 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, such as Named-Entity Recognition, Language Translation, Question Answering, and so on.

IV. Approach

We aim to answer the following research questions (RQs):

RQ1: How is domain knowledge used in user-authored scenarios?

RQ2: Given an embedding, what kind of domain knowledge can be extracted?

To answer these questions, we conduct our research with two different methods. The first method collects a user-authored scenario corpus in four directory service domains (apartment, hiking trail, restaurant, health clinic), then extracts simple domain models out of the scenarios using typed dependencies. The method is retrospective in that the corpus must exist in order to perform knowledge extraction. The second method uses 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. The 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 two approaches in detail below.

1) Domain Knowledge in User-Authored Scenarios: We investigate RQ1 by collecting and analyzing an English-language, user-authored scenario corpus in the directory services domain, which covers any software used to find or locate a thing of interest, such as an apartment or restaurant. The scenarios were collected using Amazon Mechanical Turk in an IRB-approved research study. Amazon Mechanical Turk (AMT) workers volunteered and consented to participate by accepting the human intelligence task (HIT), and then were provided a single question, e.g., "How do you find an apartment?" and asked to write a scenario that includes four elements: *steps* in the process; *goals* that the author wants to achieve; *values* or qualities the author pays attention to; and *obstacles* that could go wrong with the process, and how the author responds. The question prompts were created to reflect a variety of directory service situations, from unified services, wherein a user uses a single web application (app) to answer the question, to where a user uses a collection of unaffiliated services to forage for their answer by combining information from multiple web apps. The prompts used in this step appear in Table I. We use the *category name* of a prompt to refer to the scenarios elicited using that prompt. Eligible workers completed over 5,000 Human Intelligence Tasks (HITs), had an approval rating greater than 97%, and were located in the United States. Scenarios were rejected if they did not respond to the prompt, or if they contained more than five spelling, capitalization or grammar errors. 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. Scenarios were ranked based on their coverage of the four elements. Finally, worker identifiers were removed from the scenarios prior to analysis.

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?

TABLE I
SCENARIO PROMPTS LISTED BY CATEGORY

Next, 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 [33] based on the CoreNLP framework to identify verbs and dependent nouns in *obj* or *obl* typed dependencies. These results are reported in Section V-A.

2) Domain Knowledge in Masked Language Models: We investigate RQ2 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. 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.

The results of the above approaches will be shown in Section V-B.

V. RESULTS

We show our results for both of the research methods mentioned in Section IV. We discuss how the results may be used to extract domain models in Section VI-A.

A. Results for Domain Knowledge in User-Authored Scenarios

We present the result of scenario corpus collection and analysis. The scenario corpus was constructed by recruiting 65 distinct crowd workers to respond to 4 scenario prompts shown in Table I. Each worker may choose to answer one or more of the prompts, with each worker answering an average of 1.2 prompts. The corpus consists of 78 authored scenarios, each with a minimum of 150 words, yielding 737 sentences and 14,614 words, overall.

Table II - V 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 II
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 III
TOP-10 NOUNS RANKED BY DEPENDENCY COUNT FOR RESTAURANT FINDING SCENARIO

B. Results for Domain Knowledge from Word Embeddings using MLM

We now report 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 IV-2. In all experiments below, we used the Masked Language Model from HuggingFace, pretrained on a union of five large English corpora [18].

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 IV
TOP-10 NOUNS RANKED BY DEPENDENCY COUNT FOR HIKING TRAIL FINDING SCENARIO

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 V
TOP-10 NOUNS RANKED BY DEPENDENCY COUNT FOR HEALTH CLINIC FINDING SCENARIO

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 [38].

1) 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.

a) 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)

b) 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.

2) 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 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 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 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 extremely [MASK].
Result busy (0.0828), poor (0.0793), small (0.0523), dilapidated (0.0507), crowded (0.0467)

3) Seed Question with Inputs: includes the seed questions and an inputted response from a prior query.

a) 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)

b) 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.

VI. Discussion

We now discuss the significance of the results.

A. Domain Model Extraction

Both approaches to extract domain knowledge 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 first approach exists and has a large number of statements containing these domain knowledge, the first approach may not be easy to find the domain knowledge required to build the domain models like we do using the second approach.

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 [31]. 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 II through V, one can observe the incompleteness of domain knowledge across multiple authors. In Table II, out of 20 authors, only five authors mention "budget," while eight mention "appointments" to see the apartment In Table IV, 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, 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.

In addition, the domain knowledge extracted from user-authored scenarios can include information about actions and desired qualities. We believe this information can be used to

reason over or construct process models that include steps in a process, and desired qualities to be achieved in states within the process. Process models are complementary to class diagrams, which are topic of study in domain modeling [2], [3], [14].

B. Typed Dependency Techniques

Typed dependencies alone could be enhanced with additional techniques. For example, “place” is the fourth most-frequent noun and possibly also a hypernym, or more general concept, of the noun apartment. In contrast, the noun “area” may describe the geographic area around the apartment. Lexical databases, such as WordNet [11], may be used to check whether frequent nouns are semantically related using synsets. In WordNet, the nouns “apartment” and “place” are indirectly related via home#2 and home#1 synsets, respectively. This indirection can be difficult to discover manually and automatically, and may not be generalizable.

Similarly, the noun “one” is an English pronoun that can refer to a previously mentioned noun, called *anaphora*. Frequently used anaphora introduce ambiguity into the extracted domain model, because the noun may refer to an apartment, a budget, and so on. The pronoun can also be used to refer to the author in third-person, but this would not likely appear in the *obj* or *obl* dependency. To resolve anaphora, one can use co-reference resolution to detect and disambiguate the anaphora by linking the pronoun to the earlier noun to which it refers.

C. Seed Question Enhancement

In Section IV-2, 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 IV-2, 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.

D. The Effect of Using Intensifiers in Seed Questions

In Section V-B2, 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.

E. 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 V-B, 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 [32]. 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 [5], which was trained for scientific text, and BioBERT [19], 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 VI-A.

VII. Conclusion and Future Work

In this paper, we examined two approaches to extract domain models from a corpus and to extract domain models from word embeddings using masked language models (MLM). The

first approach is based on typed dependencies, and the second 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 [38].

We envision three prospects for future work. First, while domain models can be manually constructed using elicitation and human subjects, a generalizable, automated approach that depends on limited seed knowledge could support more advanced, automated analyses of requirements. For example, using 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). That said, we do not believe MLMs can substitute for subject matter expertise. Because the responses are statistically dependent on large corpora, they are also more likely the most obvious responses. In contrast, subject matter experts responses to queries can include domain elements situated in word contexts that are rare and infrequent. Second, the domain knowledge extraction using the MLM approach has shown that using prior inputs in the seed question templates can yield more domain knowledge, and that process may be repeated until a domain model is build to a level of satisfaction determined by the domain analyst. A future work direction is to consider a more scalable and reliable manner of measuring satisfaction, perhaps by forming specific criteria or metrics to determine when that satisfaction level is reached. For example, we may determine that satisfaction level is reached when we stop seeing new words being generated from MLMs, a concept called saturation, or we may consider using model confidence scores to try to determine when the satisfaction level is reached, although confidence levels are subject to change based on hyperparameters in the model and the underrepresentation of the training data [13]. Third, MLMs and their responses may be used to evaluate the readiness of an embedding to support more advanced domain analysis. If seed questions and seed questions with inputs, for example, do not yield rich responses, then more fine-tuning with domain-specific corpora may be needed to prepare the embedding for this domain. An advance in this third direction would ideally include generalizable metrics for evaluating embeddings for specific domains, which are not dependent on domain-specific templates. We are also interested in whether it is possible to find systematic ways to produce seed question templates that are altered to adapt to each new domain.

VIII. Acknowledgments

This research was funded in part by NSF Award #2007298.

References

[1] 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] C. Arora, M. Sabetzadeh, L. Briand, F. Zimmer. "Extracting Domain Models from Natural-Language Requirements: Approach and Industrial Evaluation," MODELS, 2016.

[3] 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.

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

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

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

[7] 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.

[8] 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.

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

[10] 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.

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

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

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

[14] 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.

[15] 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.

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

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

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

[32] 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.

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

[34] 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.

[35] C. Wang, X. He. "BiRRE: Learning Bidirectional Residual Relation Embeddings for Supervised Hypernymy Detection." 58th Annual Meeting of the Association for Computational Linguistics, pages 3630–3640, 2020.

[36] 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.

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

[38] Data and tools for our work: https://drive.google.com/drive/folders/1o7hG0N63bl5Gpuuvu5GsTFws4Hb4d4NL?usp=sharing