Towards Queryable and Traceable Domain Models

Rijul Saini
Dept. of ECE
McGill University
Montréal, QC, Canada
rijul.saini@mail.mcgill.ca

Gunter Mussbacher
Dept. of ECE
McGill University
Montréal, QC, Canada
gunter.mussbacher@mcgill.ca

Jin L.C. Guo
School of Computer Science
McGill University
Montréal, QC, Canada
jguo@cs.mcgill.ca

Jörg Kienzle
School of Computer Science
McGill University
Montréal, QC, Canada
joerg.kienzle@mcgill.ca

***Abstract*—Model-Driven Software Engineering encompasses various modelling formalisms for supporting software development. One such formalism is domain modelling which bridges the gap between requirements expressed in natural language and analyzable and more concise domain models expressed in class diagrams. Due to the lack of modelling skills among novice modellers and time constraints in industrial projects, it is often not possible to build an accurate domain model manually. To address this challenge, we aim to develop an approach to extract domain models from problem descriptions written in natural language by combining rules based on natural language processing with machine learning. As a first step, we report on an automated and tool-supported approach with an accuracy of extracted domain models higher than existing approaches. In addition, the approach generates trace links for each model element of a domain model. The trace links enable novice modellers to execute queries on the extracted domain models to gain insights into the modelling decisions taken for improving their modelling skills. Furthermore, to evaluate our approach, we propose a novel comparison metric and discuss our experimental design. Finally, we present a research agenda detailing research directions and discuss corresponding challenges.**

***Index Terms*—NLP, Machine Learning, Domain Model, Modelling Bot, Requirements Engineering, Trace Links.**

I. Introduction

Model-Driven Software Engineering is a paradigm that uses models and their transformations to build and analyze a system at different stages of software development. Domain models (class diagrams) are used during requirements analysis or the early stages of design to represent domain concepts in the form of classes, attributes, and relationships. For example, domain modelling expresses the informal requirement written in natural language (NL) "Students are enrolled in university. Students are identified by ID." with classes (Student, University), an attribute (ID) of the Student class, and an association (enroll) between Student and University classes. Since the task of building domain models is time-consuming and requires modelling skills and experience, several approaches have been proposed to assist the modellers [1] [2]. These approaches automate or semi-automate the construction of domain models from requirements written in NL. Despite the existing work on domain model extraction, some significant challenges remain unaddressed: (i) the extracted domain models are not accurate enough to be used directly in software development or for learning purposes, (ii) existing approaches do not provide any explanation of where the extracted model elements came from, and (iii) existing approaches do not provide ready-to-use interfaces for a learning environment.

To overcomes these challenges, our research goal is to *develop an automated approach to extract domain models with higher accuracy by combining NLP and ML techniques*. Our approach is motivated by the ModBud framework proposed in our previous work [3]. The aim of ModBud is to create a modelling bot for improving modelling skills of novice modellers and assisting practitioners. As an initial step, we implement our approach in the form of a tool to extract a domain model from a textual document which describes a domain in NL (English in our case). In this paper, the terms "problem description" and "case study" refer to these documents. In addition, we discuss a method to generate the trace links for each model element such as classes, attributes, relationships, and association cardinalities. These trace links connect the extracted model elements with the corresponding NL fragments in the requirements to express the rationale behind the decisions taken by our domain model extractor. Since there is no consistent comparison metric used in related work, we propose a comparison metric. The metric compares the domain models generated by our tool, the ground truth domain models (manually constructed by the authors of this paper and other researchers), and the domain models obtained from the approach proposed by Arora et al. which we treat as our baseline [2]. The main contributions of this paper are:

1) We propose an automated approach to extract a domain model from problem description using NLP and ML.
2) We instantiate the abstract ModBud framework [3] and implement our approach in a web-based prototype tool.
3) We propose a method to generate trace links for domain model elements to allow queries execution.
4) We propose a comparison metric to compare the extracted domain models with the ground truth.
5) We use three case studies to illustrate that our proposed approach achieves an accuracy of 49% relative to ground truth and improves the baseline by approximately 70%.

The remainder of this paper is structured as follows. Section II describes our automated approach. Section III briefly discusses related work and our experimental design. Section IV discusses our preliminary results. Finally, in Section V, research challenges and the broader implications of our study for the research and tool building communities are discussed.

2332-6441/20/$31.00 ©2020 IEEE
DOI 10.1109/RE48521.2020.00044

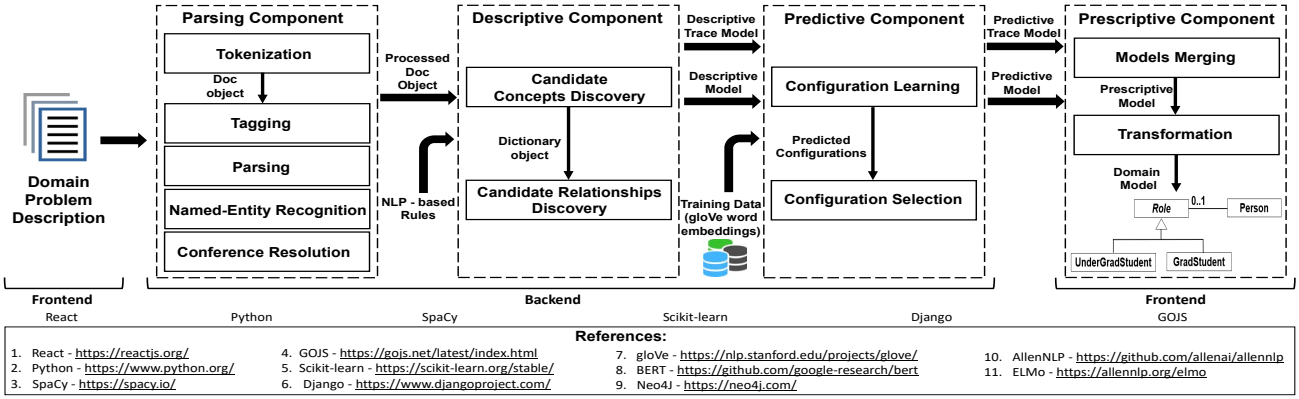

Fig. 1. Our Approach for Queryable and Traceable Domain Models

II. Automated Domain Model Extraction

In this section, we discuss our approach which combines rule-based NLP and ML to extract the domain model from a problem description expressed in NL. We also discuss our method to create trace links between domain model elements and problem description. Figure 1 summarizes our approach and method through various steps needed to visualize a domain model for a given problem description.

Parsing Component: As shown in Figure 1, in the *Tokenization* step, the problem description is split into sentences and further into words as *tokens*. The output from this step is a *Doc* object which is a sequence of tokens. Next, spaCy's linguistically sophisticated statistical models are used in *Tagging*, *Parsing*, and *Named-Entity Recognition (NER)* to predict which tag or label most likely applies in a given context. Part-of-speech tags are assigned in the *Tagging* step, dependency labels are assigned in the *Parsing* step, and named entities such as person, organization, and location are detected and labeled in the *NER* step. These steps are further integrated with the *Conference Resolution* step to find all expressions that refer to the same entity in a text. For example, in the sentence "The employees work in a company. They receive a weekly salary", the *They* token is replaced by the *Employee* token. The output from the *Parsing Component* is a processed *Doc* object holding all the meta-data of tokens.

Descriptive Component: The outcome from the above step is a processed *Doc* object which is suitable for applying rule-based NLP to extract domain concepts and their relationships. In this component, candidate domain concepts are first discovered using noun chunks extractor in *Candidate Concepts Discovery*. The extractor iterates over the *Doc* object and yields noun phrases. These noun phrases are processed against rules before they are stored in a *dictionary* object. This processing is essential as for example the noun chunk *first and last name* identified by the extractor needs to be split and merged with elements *(firstName and lastName)* based on their labels (*amod* (first), *cc* (and), *conj* (last), and *amod* (name)). These labels represent Stanford typed dependencies which provide simpler descriptions of the grammatical relationships [4].

Next, in the *Candidate Relationships Discovery* step, a syntactic dependency parser is used to navigate the tree stored in the *Doc* object. The navigation is performed as per the extraction rules where elements in the above yielded *dictionary* object are checked for *subject* or *object* based dependencies for each token which has a *verb* based dependency. This rule-based NLP approach is inspired by state-of-the-art model extraction rules [2] [5]. The outputs from these steps are the *Descriptive Model* and the *Descriptive Trace Model*. When the *Descriptive Component* is instantiated with a problem description, the *Descriptive Model* and *Descriptive Trace Model* are represented by data frames at the object level. The *Descriptive Model* abstracts the data of a problem description through various stages (from tokens to concepts and relationships) and the *Descriptive Trace Model* links these *Descriptive Model* elements with each other. In future work, these models will be persisted as knowledge graphs and used for querying to gain insights into the modelling decisions taken by our domain model extractor.

Predictive Component: In step *Configuration Learning*, we use an ML model to predict a configuration which is a refined version of the *Descriptive Model* obtained in the previous step. In this paper, we focus on the attributes of the domain model. Since for a given problem description, it is either difficult or not possible to extract the attributes and their types using rule-based NLP, we use two ML classifiers in this phase. One classifier helps in predicting whether an extracted domain concept from the previous phase is an attribute or a class. The second classifier predicts the type of attributes, such as string, integer, float, enumeration, date, and time. We created a training data set of common nouns such as name, university, and student and manually tagged them with their category - attribute or class and type of attribute. In both training and test data, noun words are represented with vectors using gloVe (see Figure 1). gloVe creates word vectors that capture meaning in vector space and takes advantage of global count statistics instead of relying just on local statistics.

The *Configuration Learning* step results in multiple predicted configurations as a given concept can be modelled

Table I. COMPARISON METRIC

Category	Sub-category	Criteria	Rationale
Classes	Class	C1: Class is present	Class is an important structural unit in a domain model.
	Super Class	C2: Class name is properly written	
	Sub Class	C3: Class name is synonym or closely related to the concept name	The presence of class with an appropriate name helps in identifying the domain concepts.
	Enumeration Class		
	Association Class	Same as C1, C2, and C3	Same as for C1, C2, and C3.
		C4: With right participating classes	The presence of an association class is based on the instances of other participating classes.
Relationships	Association	C5: Relationship exists with right pair of classes	Relationship establishes the relation between the instances of participating classes.
		C6: Role names are present	Role names help in navigating the domain model.
		C7: Relationship has correct direction if unidirectional	Direction helps in navigating the domain model.
	Generalization	Same as C5, C6, and C7	Same as for C5, C6, and C7.
		C8: Source class and target class are correct	Source class and target class are not interchangeable for these types of relationships.
	Aggregation	Same as C5, C6, C7, and C8	Same as for C5, C6, C7, and C8.
	Composition	Same as C5, C6, C7, and C8	Same as for C5, C6, C7, and C8.
Attributes	Attribute	C9: Attribute name is correct	It helps in identifying the data held by the instances of class.
		C10: Attribute type is correct	Same as for C9
Cardinalities	Zero to one	C11: Multiplicities are present and correct.	Helps in identifying the instances of one class associated with the instances of another class.
	One to many		
	Zero to many		
	Number based		

differently, e.g., "part-time student" and "full-time student" can be modelled by using the *Player Role* pattern or by simply defining a "StudentRole" enumeration type. Both configurations are correct and restrict the creation of different objects for the same student each time the student changes his role from "part-time" to "full-time" or vice-versa. In the step *Configuration Selection*, confidence estimation techniques are used to determine the reliability of individual configurations and select the configuration with the highest confidence score. The outputs from this component are *Predictive Model* which predicts a possible configuration of the domain model and *Predictive Trace Model* which links the *Predictive Model* elements to the *Descriptive Model* elements.

Prescriptive Component: In step *Models Merging*, the *Predictive Model* and the *Descriptive Model* are merged to generate the *Prescriptive Model* which defines the domain model. Finally, in step *Transformation*, the *Prescriptive Model* is transformed into a form suitable for visualization as a domain model (class diagram).

III. RELATED WORK AND EXPERIMENTAL DESIGN

In this section, we first discuss the comparison metric, related work and our chosen baseline, as well as the ground truth domain models used in our paper. Next, we present the three research questions and methodologies to answer them.

Comparison Metric: We propose a novel comparison metric as shown in Table I that makes it possible to consistently measure the semantic closeness of a domain model compared to another domain model, i.e., the ground truth. If a model element in the model that is being compared matches a model element in the ground truth, then the domain model gets points (class [+4], relationship [+0.5], cardinality [+0.5], and attribute [+0.5]). For example, for an informal requirement "A university consists of many departments", a domain model obtains a total score of 9 with two classes (University, Department) and a composition relationship with zero to many (0..*) cardinality between University and Department. If there is an irrelevant model element that should not be in the domain model, then a deduction is applied to the domain model (class [-1], relationship [-0.25], and attribute [-0.25]). The final score is normalized on a scale of 100.

Related Work and Baseline: Many approaches have been proposed to extract models from requirements or to assist modellers [6] [7]. Robeer et al. propose an approach to automatically extract conceptual models from user stories [5]. However, their approach requires user stories in a particular format which does not meet our criteria of processing problem descriptions in free-form text. Also, Perez et al. [1] present a modelling bot that interprets the users' inputs in the form of simple sentences in NL and builds a meta-model. In contrast, our approach is based on the vision of creating a platform which can take the whole problem description at once and generate its corresponding domain model. The approach by Arora et al. [2] extends the state-of-the-art model extraction rules with complementary rules from the information retrieval literature to develop a model domain model extractor. Their approach is closely related to our work as we also exploit such rules and implement them in the form of a prototype tool. Also, their approach has been evaluated for industrial case studies and by domain experts. Moreover, their implementation is available and replicable on our system. Therefore, we consider the approach by Arora et al. [2] as our baseline.

Ground Truth Domain Models: Modelling experts construct these models while performing modelling exercises as part of several curricula at universities. Since these models represent ideal solutions of the corresponding domain problems, the final score for these models is 100% after normalization.

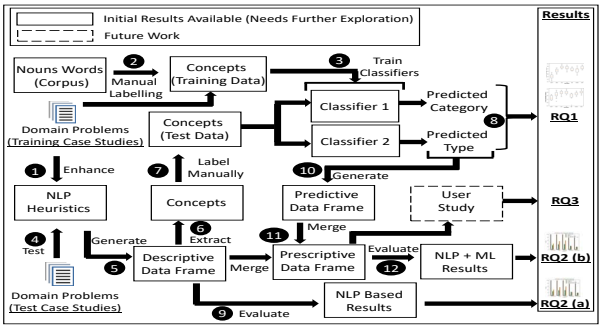

Fig. 2. Experimental Design

Research Questions: In line with our research goal, we formulate the following three research questions to study and evaluate the performance of our approach and tool.

RQ1) How do different machine learning models perform in predicting the attributes and their types?

Motivation In some cases, extraction of attributes is possible using rule-based NLP, e.g., "name" attribute from an informal requirement "Employee is identified by name". However, prediction of attributes and their types often requires the semantic and context information of concepts which goes beyond what is possible with rule-based NLP. For example, "name" concept is always used as an attribute with "string" type and "address" concept could be used as an attribute (if only one type of address in problem description) or as a class (multiple types of addresses such as permanent, temporary, and mailing).

Methodology We use gloVe to generate word embeddings by aggregating the global word-word co-occurrence matrix from a corpus. As shown in Figure 2, in step (1), we develop the rule-based NLP approach using case studies written in NL (training data set). In step (2), we collect concepts from the corpus of noun words as well as from training case studies. These concepts may represent a class or an attribute. Therefore, we manually label them with an appropriate concept category and attribute type. This labeled data is then used to train two classifiers in step (3). One classifier is used to predict the concept category and the other classifier is used to predict the attribute type. In step (4), we use a different set of case studies (test data set) and generate a descriptive data frame for each as shown in step (5). Next, in step (6), we extract concepts and label them manually in step (7) with a suitable category (attribute or class) and if the concept is an attribute then we also label its type. Finally, in step (8), predicted category and predicted type are used to evaluate the classifiers' accuracy and answer this research question.

RQ2) How accurate are the extracted domain models using our approach - a) NLP and b) NLP + ML?

Motivation Due to limitations in existing approaches such as low accuracy of extracted domain models or the necessity of transforming problem descriptions into some standard or structured format, it is important to build a robust solution that can extract domain models from a problem description written in free-form text. To evaluate the accuracy of our approach, it is imperative to evaluate its performance when it uses only rule-based NLP and when it combines NLP and ML.

Methodology The descriptive data frame from step (5) is evaluated using the comparison metric to generate results (NLP based) as shown in step (9). Next, in step (10), the predictive data frame is generated on the basis of the results obtained in step (8). Furthermore, the descriptive data frame and the predictive data frame are merged in step (11) to generate the prescriptive data frame. We evaluate the prescriptive data frame in step (12) using the comparison metric to generate results (NLP + ML). These results (NLP and NLP + ML) are compared with the scores of the ground truth domain model and baseline to answer research question 2(a) and 2(b).

RQ3) How effective and useful are the results of our tool to improve modelling skills?

Motivation We envision to use our automated approach in the form of a modelling bot which can help students and novice modellers in improving their modelling skills. Before a modelling bot can be employed in a learning environment, it is essential to evaluate the usefulness aspect of the tool by determining how reliable are the generated domain models. Also, it is crucial to determine how accurate domain models have to be before they can be used to guide novice modellers. Moreover, the effectiveness of the tool needs to be assessed in terms of the results of queries which rely on trace data frames as they bridge the gap between a problem description and its corresponding domain model. The tool would allow modellers to execute queries, e.g., "why has the StudentRole class been created?" on the generated domain model and help them in tracing the reason of this modelling decision through different trace data frames to the problem description. We further envision scenarios where the generated domain model is hidden from modellers and the modelling bot helps them in building the domain model and improving their modelling skills by guiding them through these trace data frames.

Methodology We plan to conduct a user study to assess the effectiveness and usefulness of our tool. In this study, there will be two groups of participants - experienced modellers or educators and novice modellers or students with no modelling experience. The study is composed of three parts. The first part focuses on information about modellers and demographic questions on their modelling experience. The second part asks the experienced modellers to complete a questionnaire after using the tool and evaluating the generated domain models and results of their queries. The aim of this questionnaire is to assess the usefulness aspect of our tool. The third part invites novice modellers and provides them with some basic knowledge about domain modelling. Next, novice modellers are split randomly into two groups - control group and study group. The control group completes a questionnaire with questions on domain modelling and a domain modelling exercise without using our tool. The study group will use the tool to develop their modelling skills and complete the same questionnaire and exercise. The groups' responses to the questionnaire and their domain models will be evaluated and compared to discern whether our tool is effective in improving modelling skills.

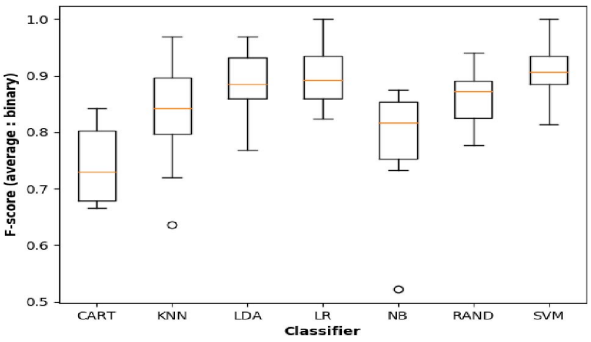

Fig. 3. Machine Learning Models Comparison (Concept Category)

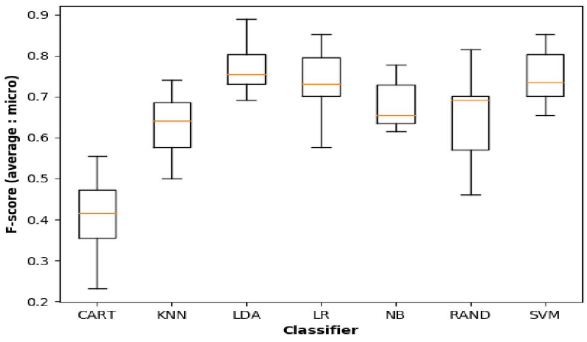

Fig. 4. Machine Learning Models Comparison (Attribute Type)

IV. Preliminary Results

In this paper, we use four case studies to develop our approach and a different set of three case studies to evaluate it. These case studies represent NL problem descriptions of systems, e.g., Election Management and Airline Ticket Reservation. Typically, first or second year undergraduate students are expected to construct the corresponding domain models in approximately 30 minutes. By applying our approach and performing the experiment, we set out to better understand (i) performance of different machine learning models for attributes, (ii) accuracy of the extracted domain models using our approach, and (iii) effectiveness and usefulness of the results of our approach. In this section, we derive and discuss initial answers to our first two research questions (RQ1, RQ2).

Performance of different machine learning models for attributes: We consider different supervised ML classifiers and evaluate their F-score after tuning their hyperparameters and using k-fold cross-validation techniques. The classifiers are Decision Tree (CART), k-nearest neighbors (KNN), Linear Discriminant Analysis (LDA), LR (Logistic Regression), Naive Bayes (NB), Random Forest (RAND), and Support-Vector Machines (SVM). We compare their performance using a boxplot graph which is a standardized way of displaying the distribution of data (minimum, first quartile (Q1), median, third quartile (Q3), and maximum). Also, some points represented as dots determine the outliers. As shown in Figure 3, the classifiers perform binary classification (class and attribute). SVM achieves the highest median F-score (0.91) and it has the least variation (maximum - minimum) as compared to other good performing models (LR, RAND). Likewise, in Figure 4, models perform multi-class classification (date, enumeration, float, integer, string, and time). LDA achieves the highest median F-score (0.78) and its variation is comparable to SVM with second highest median F-score (0.75) and NB (least variation). Consequently, we use the SVM classifier to predict the category of concepts and filter out the attributes. For these attributes, we then use the LDA classifier to predict their types. Also, we observe that LDA achieves higher accuracy (86% to 91%) for enumeration, time, and string types, while lower accuracy (67% to 70%) for date and float types due to a larger variety of data encodings possible for the latter types.

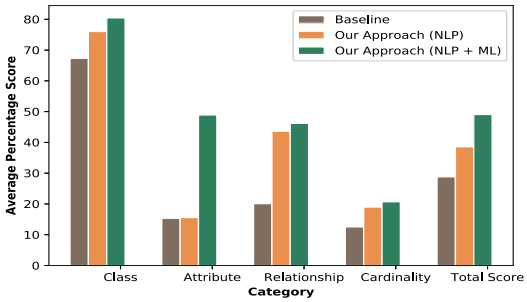

Fig. 5. Evaluation of Extracted Domain Models (3 Case Studies)

Accuracy of the extracted domain models using our approach: Figure 5 illustrates the average percentage scores achieved by the baseline and our proposed approach (NLP and NLP + ML) for the 3 case studies using our comparison metric. These scores are relative to the scores of the ground truth domain models (100% in each category). The lower scores of the baseline could be attributed to the extracted noun chunks which are not always at the right level of granularity and to a large number of false positives in concepts and relationships. The total average scores of baseline and our approach (NLP and NLP + ML) with respect to ground truth are 28.78%, 38.52% and 49% respectively. The results are encouraging but there is room for improvement.

In this paper, we minimize the internal threat of validity by considering the case studies and their corresponding ground truth domain models used by researchers and educators other than authors. In the future, more case studies need to be considered to evaluate the performance and analyze the limitations of our approach. Moreover, we envision to develop the complete system and conduct a user study to answer the third research question (RQ3).

V. Research Directions and Implications

Our research goal is to build a modelling bot to assist modellers. We identify the following research challenges while envisioning a modelling bot and discuss them in this section.

Accuracy of generated domain models: Our tool extracts a domain model from a given problem description written in

NL. The results of queries executed on the knowledge base and the teaching strategies employed by a modelling bot to help novice modellers largely depend on the accuracy of extracted domain models and trace models. Some challenges which we identify in this research direction are:

a. Ambiguities in NL based requirements: NL is universal and widely used to express requirements, but unfortunately also inherently ambiguous. Researchers have explored several definitions of ambiguities and discuss the problem of detecting ambiguities in requirements expressed in NL and techniques to deal with them [8] [9] [10] [11]. Some techniques, e.g., obtaining more contextual information can be applied with our approach where the modelling bot may check with a human user when there is insufficient information provided in the problem description to be processed by the rule-based NLP.

b. Word embeddings: Word embeddings give us a way to use an efficient and semantic-laden word representations in which similar words have a similar encoding. In this paper, we use gloVe as a word embedding model to generate word vectors for the candidate concepts. However, further investigation is required in considering the whole sentences of these candidate concepts to capture their context. In addition, other dynamic embedding methods such as BERT and ELMo (see Figure 1) need to be investigated.

c. Data quantity and quality: Applications, especially those using ML, not only need large amounts of data to perform optimally but also require a balanced data set primarily in the context of supervised machine learning involving two or more classes. In this paper, we use traditional supervised machine learning algorithms to predict attributes and their types. Other types of algorithms, e.g., unsupervised, semi-supervised, and reinforcement learning could be explored. Also, more labeled data is required in the form of domain model elements while working with supervised learning methods.

Querying a knowledge base: Our approach uses trace models which could be stored as knowledge representation graphs. These graphs have been used widely to represent requirements, e.g., Schlutter and Vogelsang developed an NLP based pipeline to transform a set of heterogeneous NL requirements into a knowledge representation graph [12]. To use a modelling bot in learning environments, it is imperative that the queries can be executed efficiently and their results can be obtained in real-time. Therefore, graph database management systems, e.g., Neo4j and platforms for question answering, e.g., AllenNLP (see Figure 1) need to be explored.

Traceability of modelling decisions: Cleland-Huang et al. [13] provide detailed reviews of the state-of-the-art on requirements traceability and describe the three areas which must work synergistically to automate the trace creation process towards the goal of eradicating manual traceabilty effort. The three areas are trace creation, trace maintenance, and trace integrity. For trace creation in a modelling bot, trace links should be automatically generated between model elements, the descriptive model, and the predictive model. For trace maintenance, trace links should be updated automatically when there is a change in the problem description, descriptive model, or predictive model. Finally, trace integrity is concerned with correctness and completeness of the trace links to assess if they can be trusted to gain insights into the modelling decisions.

Bot for teaching modelling: Zarras et al. investigate today's publicly available ML services to understand the extent to which they can be used to pass image-based Turing Tests [14]. In this paper, we propose the design of a user study to gain a better understanding of how effective and useful our approach is for modellers. More empirical studies are required to study the extent to which a modelling bot can imitate the teaching style of an educator (experienced modeller) so that its behavior remains indistinguishable from the educator's behavior.

To conclude, domain modelling plays a significant role in transforming the informal requirements expressed in NL to more precise and analyzable specifications. In this paper, we describe our automated approach to overcome the challenges in domain modelling and implement our approach in the form of a web-based tool. We further present research questions and methodologies to evaluate them. Finally, we discuss the preliminary results and present the research challenges and the prospects of research endeavors in this direction.

REFERENCES

[1] S. Pérez-Soler, E. Guerra, J. de Lara, and F. Jurado, "The rise of the (modelling) bots: Towards assisted modelling via social networks," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2017, pp. 723–728.

[2] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Extracting domain models from natural-language requirements: approach and industrial evaluation," in *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, 2016.

[3] R. Saini, G. Mussbacher, J. L. C. Guo, and J. Kienzle, "Teaching modelling literacy: An artificial intelligence approach," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, Sep. 2019, pp. 714–719.

[4] M.-C. De Marneffe and C. D. Manning, "Stanford typed dependencies manual," Technical report, Stanford University, Tech. Rep., 2008.

[5] M. Robeer, G. Lucassen, J. M. E. van der Werf, F. Dalpiaz, and S. Brinkkemper, "Automated extraction of conceptual models from user stories via nlp," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*. IEEE, 2016, pp. 196–205.

[6] O. S. Dawood *et al.*, "From requirements engineering to uml using natural language processing–survey study," *European Journal of Engineering Research and Science*, vol. 2, no. 1, pp. 44–50, 2017.

[7] F. B. Aydemir and F. Dalpiaz, "Towards aligning multi-concern models via nlp," in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2017, pp. 46–50.

[8] D. M. Berry and E. Kamsties, "Ambiguity in requirements specification," in *Perspectives on software requirements*. Springer, 2004, pp. 7–44.

[9] A. K. Massey, R. L. Rutledge, A. I. Antón, and P. P. Swire, "Identifying and classifying ambiguity for regulatory requirements," in *2014 IEEE 22nd international requirements engineering conference (RE)*. IEEE.

[10] A. Ferrari, G. Lipari, S. Gnesi, and G. O. Spagnolo, "Pragmatic ambiguity detection in natural language requirements," in *2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*. IEEE, 2014, pp. 1–8.

[11] V. Gervasi and D. Zowghi, "On the role of ambiguity in re," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2010, pp. 248–254.

[12] A. Schlutter and A. Vogelsang, "Knowledge representation of requirements documents using natural language processing." RWTH, 2018.

[13] J. Cleland-Huang, O. C. Gotel, J. Huffman Hayes, P. Mäder, and A. Zisman, "Software traceability: trends and future directions," in *Proceedings of the on Future of Software Engineering*, 2014, pp. 55–69.

[14] A. Zarras, I. Gerostathopoulos, and D. M. Fernández, "Can today's machine learning pass image-based turing tests?" in *International Conference on Information Security*. Springer, 2019, pp. 129–148.