An Interactive Tool for Goal Model Construction using a Knowledge Graph

Shahin Abdoul Soukour, William Aboucaya, Nikolaos Georgantas

An Interactive Tool for Goal Model Construction using a Knowledge Graph

Shahin Abdoul Soukour[1 [0009−0005−5026−712X]], William Aboucaya[1 [0000−0003−2413−6968]], and Nikolaos Georgantas[1 [0000−0001−5704−4889]]

Inria, Paris, France {abdoul-shahin.abdoul-soukour,william.aboucaya,nikolaos.georgantas}@inria.fr

Abstract. [**Context and motivation**] The goal model is an essential model in Goal-Oriented Requirements Engineering. It is used to describe the system's goals using a hierarchical structure in which high-level goals are refined into more specific ones.
[**Question/problem**] Constructing a goal model for a new application can present challenges, demanding considerable time and effort. Although there have been attempts to automate or semi-automate the construction of goal models, these tasks remain complex and manual.
[**Principal ideas/results**] This paper presents an interactive graphical tool that leverages a domain Knowledge Graph (KG) to assist the application designer in creating goals derived from this knowledge, thereby facilitating the creation of goal models. We use semantic similarity measurement and Natural Language Inference (NLI) to effectively extract and align triples from the KG with the high-level initial goals formulated by the application designer. The extracted triples undergo sentiment analysis and Graph-to-Text (G2T) generation to build meaningful subgoals. Nevertheless, processing KGs with Natural Language Processing (NLP) techniques can be a lengthy process. We introduce a restriction-based approach to bound the exploration of the KG to the most promising nodes. By tuning KG exploration bounds while using our tool in a case study, we analyze the trade-off between the quality of the resulting goal model and time performance, which is a key factor for an interactive approach.
[**Contribution**] Our paper highlights the relevance of our restriction-based approach to information retrieval in KGs to facilitate goal model generation.

Keywords: Goal modeling · Goal-Oriented Requirement Engineering · Requirement Engineering · Knowledge Graph · Natural Language Processing · Natural Language Inference · Graph-to-Text

1 Introduction

Goal-Oriented Requirements Engineering (GORE) [17,18,19] is a specific approach to Requirements Engineering (RE). RE is the first step in the design process of a software system. GORE's main focus is on identifying, capturing,

and modeling stakeholders' goals for such a system. Various goal-oriented modeling methods have been proposed, such as [4,12,18]. The **goal model** is the central model of the GORE methods. It is a structural approach that enables goals to be defined and analyzed, and helps to understand the stakeholders needs. It provides a representation of goals in an explicit way, where high-level goals (strategic, global) are refined (or decomposed) into lower-level ones (operational, local, design-specific) until a clear understanding of the system is reached. Ultimately, the leaf goals of the goal model are elementary goals and are considered as specifications of software components. Building a goal model for a new application is a complex and challenging task, requiring both time and effort from the application designer [18]. Complexity comes from both identifying the goals of the new system and organizing them into a goal model.

A few approaches have proposed to generate a goal model in an automatic or semi-automatic manner by extracting goals from a small text corpus. Most approaches apply Natural Language Processing (NLP) techniques on various text sources. Das *et al.* [8] propose an end-to-end framework that uses several NLP techniques in order to transform automatically a set of unstructured natural language requirement specifications into goal models, supporting goal refinement, associations among goals, resources and softgoals. In this case, requirements are already provided by stakeholders; the focus is on organizing them in a structured way. Casagrande *et al.* [6] collect abstracts of research publications related to a specific system category. They then use these texts to extract goals by looking for text parts containing goal-related keywords; from these goals a goal taxonomy is created. However, other text parts that may contain relevant domain-specific information for creating a goal are not considered. Both of the above approaches attempt to build a complete goal model from the reference text sources. The application designer only gets involved at the end of the process to check the goal model and make any necessary changes. Recent research has considered Large Language Models (LLM) in the field of RE, offering the potential to drive RE processes [2]. They are also employed to generate and complete goal models. Nakagawa *et al.* [21] presented an approach for semi-automatically generating a goal model based on a LLM within a MAPE-K loop mechanism (an autonomic loop for implementing self-adaptive systems) that iteratively refines the goal model. However, the information extracted from the LLM may lack precision for a particular domain. The intervention of domain experts is necessary to effectively validate and refine the goal model.

In contrast to these solutions, our approach, which was first published in [1] and is updated with new results in the present paper, proposes using domain Knowledge Graphs (KG) to interactively assist the application designer in creating and refining their goals into a goal hierarchy, inspired from the KG. We consider that domain KGs are extremely valuable resources for developing goal models, as they can help identify relevant facts on a given topic. Goals describe a condition of the target system that should be achieved or avoided [18]. Relevant facts extracted from KGs can inspire target conditions related to goals [1].

To efficiently explore a domain KG, our method uses NLP techniques based on pre-trained language models. Our approach identifies areas of interest inside a KG by measuring semantic textual similarity between a designer's high-level goal and KG triples, and uses Natural Language Inference (NLI) to determine textual entailment relations between triples and the high-level goal. Such entailment relations can emulate relations between a goal and potential subgoals. Additionally, sentiment analysis is performed to detect positive and negative connotations in triples, implying target conditions that should be achieved or avoided, respectively. Finally, Graph-to-Text (G2T) generation is employed to convert triples into coherent text for facilitating the designer in their creation of new goals. Through step-by-step interaction with our solution, the designer can refine their initial goals into a goal hierarchy from the KG, which may be complemented by data from other sources to create a complete goal model.

An initial version of our approach was published in [1]. In the present paper, we report on the development of an interactive graphical tool that implements our approach. This tool enables real users to use our approach and incorporates several small improvements to our solution, pointed out in Sections 3 and 4, which present our Method and Evaluation, respectively. Nevertheless, most importantly in this paper, we enhance our solution with the capability to bound the exploration of a KG to the most promising triples. This aims at accelerating the search inside a KG, thus improving the time performance of our approach, which is a key feature for an interactive tool. More specifically, we constrain the maximum depth of exploration of the graph for each information retrieval process and the number of neighbors from a given triple explored at each step of the exploration process.

Based on this approach, we assess the trade-off between the quality of the resulting goal model and time performance. In particular, we use our tool to design of a flood management system similar to the one proposed in [1]. More specifically, we identify the share of additional (ir)relevant information retrieved by the more exhaustive versions of our graph exploration algorithm based on beam search, as well as the time-performance cost induced.

Code and KG used to produce the results of this paper are available at `https://github.com/ShahinAbdoulSoukour/Knowledgeable_Goal_Modeling`

2 Background

2.1 Natural Language Inference

Natural Language Inference (NLI), also known as **Textual Entailment Recognition** (TER), involves examining a pair of sentences, usually called a *premise* and a *hypothesis*, and determining the relationship between them. The relationship can be classified in one of three categories: entailment (the hypothesis can be inferred from the premise), contradiction (the hypothesis directly contradicts or opposes the premise) or neutral (if the relationship between the premise and the hypothesis is neither an entailment nor a contradiction). Several

methods have been proposed to deal with NLI tasks [30]. In this context, we will focus on language models specifically trained for this purpose. This approach typically uses transformer-based models, such as BERT [9] or RoBERTa [20]. These models are well-suited for NLI tasks due to their contextual understanding of a sentence or a pair of sentences, and often outperform other methods. Multiple datasets have been proposed for training and evaluating models on this task, such as the Stanford NLI (SNLI) which includes 570k human-written English sentence pairs [3], Multi-genre NLI (MNLI) which includes 433k sentence pairs and covers from ten distinct genres spoken and written text (fictions, travel guides, reports, etc.) [32] or Adversarial NLI (ANLI) [24] which contains three rounds of sentence pairs (R1, R2, R3), with each round becoming progressively difficult. This dataset is designed to be more challenging than the previous ones and focuses on crafting difficult examples. In this paper, NLI tasks are performed using a RoBERTa$_{LARGE}$ model trained using the SNLI, MNLI, ANLI and FEVER-NLI [23] corpuses[1]. Performances of this model on the different datasets used for training are available in [24].

2.2 Graph-to-text generation

Graph-to-Text generation (G2T) aims at generating fluent texts from knowledge stored as a graph. The most commonly used evaluation criterion for G2T models is the similarity of the generated text to a reference description of the KG written by a human. This is typically measured using metrics such as BLEU [25] or BERTScore [34]. However, metrics more specially related to G2T tasks have also been proposed to evaluate the *factual faithfulness* of the text generated compared to the initial data, such as Data-QuestEval [27] or FactSpotter [33]. Currently, most state-of-the-art models for G2T are based on a Transformers architecture [16,29]. The pre-training and evaluation of these models is generally performed using datasets that contain pairs consisting of series of facts expressed as a graph and a text summarizing its content. Examples of such datasets include DART [22] or WebNLG [7]. In this paper, G2T tasks are performed using a T5$_{BASE}$ [26] model trained by using the WebNLG'20 dataset[2]. Performances of this model on multiple metrics, including the factual faithfulness of the generated texts in relation to the original data, are available in [33].

2.3 Goal model evaluation

Goal model evaluation is an important part of GORE. This process involves evaluating the quality of goal models. Most goal model evaluation techniques are either *qualitative* or *quantitative* [18]. Qualitative evaluation means manually inspecting the goal model to identify potential issues, inconsistencies (consistency checking) or improvements. This type of assessment relies on expert judgement.

[1] Model available at `https://huggingface.co/ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli`

[2] Model available at `https://huggingface.co/Inria-CEDAR/WebNLG20T5B`

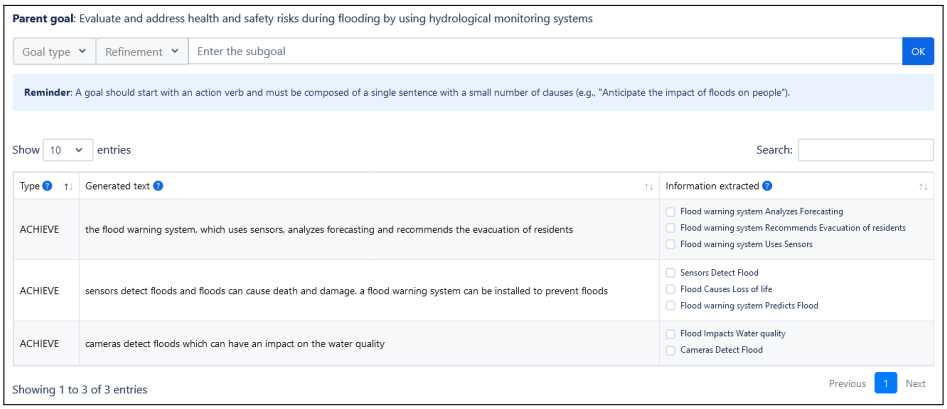


Fig. 1: Screen capture from the proposed tool: Creation of a subgoal based on information related to its parent goal.

And the quantitative evaluation involves using metrics to assess the quality and effectiveness of the goal model such as the goal model complexity (e.g., number of goals, depth of goal decomposition) or goal achievement (e.g., percentage of goals achieved) [13]. In the case of automatic or semi-automatic goal model generation, both types of evaluation can be used for specific use cases, in particular to compare the lists of goals generated by a tool focused on goal model generation with those obtained and validated manually by an expert [6,31], or to use feedback from the expert to improve the goal model itself [21]. On the other hand, *formal verification* can be used to formalize the goal model using a mathematical or logical representation to analyze and ensure the correctness of the goal model [5,18,19]. It provides a high level of assurance about the correctness of goal models. However, it can be complex and time-consuming. It requires, nevertheless, knowledge of formal methods.

3 Method

We elaborate in this section our improved method for assisting an application designer in building a goal model for a new application by leveraging a pre-existing domain KG. Our method takes the form of an interactive tool (user interface is presented in Figures 1 and 2). The designer can define an initial high-level goal for the development of a new application. After exploring the KG, the tool provides one or more texts that analyze or refine the submitted goal. These texts derived from the goal can be used or serve as inspiration to create one or more subgoals. Our approach comprises the following steps (see Figure 3), which are quite similar to our previous approach [1]. Indeed, we improved the KG exploration algorithm to emphasize both *depth limitation* and *beam search* in the entailment exploration process. It adapts based on the entailment score, expanding the search when entailment improves and stopping when it decreases.

Fig. 2: Screen capture from the proposed tool: Visualization of an example goal model.

It leverages beam search to explore the most promising triples within a specified beam width.

1. **Triple inference.** Inference rules[3] are applied to identify the significance of triples, focusing particularly on inheritance relationships, addressing cases where the NLI fails to link the fact expressed by the triple, despite its potential contribution to the goal. Using inference rules can complete the KG.
2. **Formulate an initial goal and provide parameters.** Create an abstract, non-composite goal. Divide composite goals into multiple elementary goals. Then, provide the *maximum depth* to limit the depth of exploration and the *beam width* for exploring only the top k triple neighbors of each triple.
3. **Search for anchor triples in the KG.** In this step, we identify relevant triples in the KG based on the formulated goal and parameters. Querying the KG can be challenging, requiring knowledge of SPARQL and understanding of the KG's structure. *Keyword search* can be also used, but it requires exact matches to return results, necessitating refinement of search criteria. To address this issue, our approach employs SBERT [28] to compute the semantic textual similarity between each triple of the KG and the expressed goal. We only consider the best results, i.e., the triples whose similarity with the goal is above $c \times best\ score$, with $c \in [0, 1]$ a confidence parameter[4]. If

[3] An example is the rule "IF *subject predicate parent* AND *child rdf:type parent* THEN *subject predicate child*"

[4] Higher values of c enhance the quality of the retrieved triples while reducing their number.

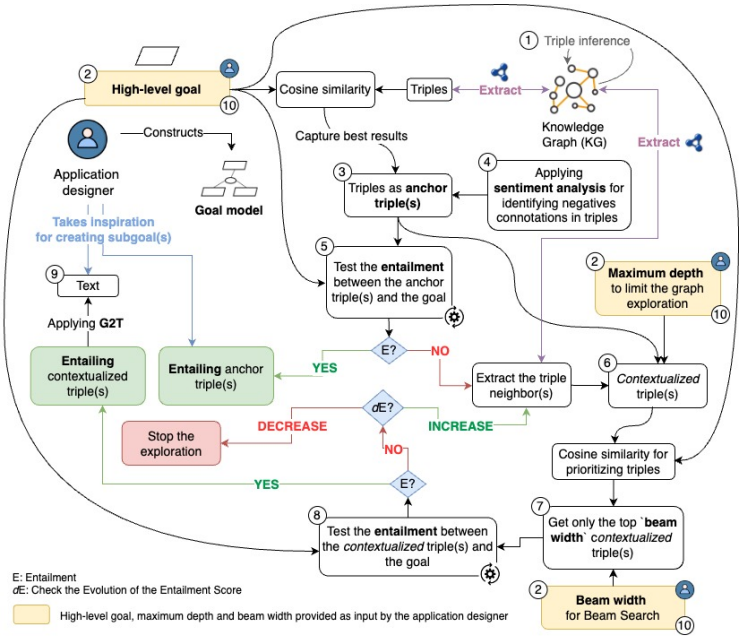


Fig. 3: Steps of our approach to facilitate goal models generation using a KG

the number of triples found is lower than an arbitrary value[5], we reduce the confidence parameter to a second threshold c_2 to capture more triples. These triples are considered as **anchor triples** in the KG.

4. **Identify connotation in anchor triples by applying sentiment analysis.** We perform sentiment analysis using a RoBERTa model[6]. Indeed, this step involves determining the sentiment of anchor triples to distinguish between facts to achieve and facts to avoid.
5. **Test entailment between anchor triples and the goal.** NLI is used to evaluate entailment between each anchor triple as premise and the submitted goal as hypothesis. If an anchor triple entails the goal, the fact expressed by the triple can be considered as contributing to the goal.
6. **Add context to non-entailing anchor triples if there is no entailment.** In cases where there is no entailment between an anchor triple and the goal, a possibility is that an implicit entailment might still exist depending on a context which has not been expressed. To tackle this, each anchor triple

[5] Here, we use 4 as the arbitrary parameter. However, this value can be increased or decreased depending on whether it is more important to provide a high number of data or only highly relevant resources.

[6] Available at `https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest`

is concatenated with its different neighbor triples to form a *contextualized triple*.

7. **Beam search for context triples.** To identify potentially relevant context triples, the semantic textual similarity between the goal and the different versions of the contextualized triple is computed.We then select the β context triples with the highest similarity to the goal, with β equal to **beam width**, and discard the other neighbor triples as irrelevant. We choose this criterion as computing the semantic textual similarity is less time-consuming than the entailment detection, and allows us to perform a beam search, which is more time-effective than an exhaustive exploration of the graph.
8. **Test the entailment between the contextualized triples and the goal.** We retain the one with the highest entailment score. If entailment is achieved, proceed to the next steps. If entailment does not hold, repeat the process (**Steps 6** to **8**) by adding another context triple to the already contextualized triple. Continue this iterative contextual enrichment as long as the entailment score increases. Stop the process if entailment is reached, if the entailment score decreases, or if the number of context triples is equal to the *max depth* parameter.
9. **Return entailing triples and create subgoals.** G2T is applied to transform the entailing contextualized triples into meaningful texts that help the application designer in creating subgoal(s). The prefix "[ACHIEVE]" is added to texts resulting from anchor triples with positive or neutral connotations, indicating that the information in the text contributes to the achievement of the goal. Conversely, the prefix "[AVOID]" is added to texts resulting from anchor triples with negative connotations, indicating obstacle or undesirable result to be avoid in order to achieve the goal.
10. **Repeat the process for refining the goal model.** The interaction between the application designer and the tool can be repeated several times, giving the designer the opportunity to refine and extend the goal hierarchy if necessary by adding new goals from other sources, or by drawing inspiration from the relevant triples (or facts), as provided by our tool by adjusting the two parameters. Indeed, the designer can reviews the triples, focusing on those that contribute to the high-level goal and align with the context of the application.

4 Evaluation

This section presents the evaluation process of our method. Indeed, we have evaluated our method by applying it to the creation of a goal model for a flood management and risks mitigation system. We have implemented our method as a software tool available through a graphical user interface and uses a domain

GM	max depth	beam width	NET	NSET	M10	NLG	MD	T (sec)	R (%)
1	1	3	9	2	1	1	1	10.715	22.222
2	3	1	44	20	11	5	5	11.858	**45.454**
3	3	2	50	18	10	4	5	13.283	36.000
4	3	3	49	19	10	4	5	13.735	38.775
5	3	5	46	19	10	4	5	13.290	41.304
6	5	3	54	20	11	5	5	13.670	37.037
7	1	1	9	2	1	1	1	10.460	22.222
8	99	99	75	29	14	7	5	16.941	38.666

Table 1: Goal model construction performance and relevance metrics

KG related to flood management from [1][7], which we enriched by adding new triples. This tool facilitates the generation and visualization of goal models.

The KG is composed of 90 triples (61 nodes linked by 28 different types of edges). It provides an overview of the basic principles of flood management and risks mitigation. It covers the possible causes of floods, techniques for detecting them, the anticipated effects of floods on residents and infrastructure, and measures to prevent or to minimize these impacts. We use a configuration of our tool with confidence parameters $c = 0.85$ and $c_2 = 0.65$ as the arbitrary thresholds for semantic similarity.

To evaluate the quality and effectiveness of our tool, we used it to construct eight goal models (**GM**). In this process, we adjusted two core parameters (**max depth** and **beam width**) to control the depth of exploration within the KG and to filter relevant triples (triple neighbors) using beam search. We performed the experiment for multiple values of each of these two parameters, including a baseline with the two parameters set to 99 to replicate the behavior of an unconstrained version of the tool. Each generated goal model progressively refined the initial high-level goal, "*Anticipate the impact of flooding on people*" allowing us to capture variations in structure and relevance across different configurations (see Table 1). After constructing these goal models, we extracted the coverage of relevant triples for each goal model, particularly focusing on the number of entailing triples (i.e. triples that can contribute to the high-level goal) provided by our tool (**NET**), the number of entailing triples selected by the designer for goal refinement (**NSET**), the number of direct or indirect refinements of the high-level goal (called **M10** in [10]), and the number of leaf goals (**NLG**). Moreover, we measured the depth of the goal hierarchy (called **MD** in [11]) and also the execution time for each refinement in a goal model. We use **NET** and **NSET**

[7] *N.B.:* After publication of our previous work [1], a bug was identified which led the tool to behave as if the **max depth** parameter was set to 1, even though this parameter had not yet been implemented. The intended behavior was for the tool to function as if **max depth** was set to $+\infty$. This bug has since been fixed, and the tool now correctly explores the graph. However, due to this past issue and the differences in the graph used for knowledge extraction, results – particularly time-based metrics – should not be compared between the two papers.

metrics to quantify both the number of facts highlighted by the tool and, out of these, the number of facts which are relevant to help refine high-level goals into more specific ones. Then, we calculated the mean execution time (**T**). Next, we calculated the relevance (**R**), which indicates how effectively the selected entailing triples contributed to the refinement of the goal model (see the equation 1). For this experiment, we used an Nvidia 3080 RTX GPU with 32 GB RAM.

$$R = \left(\frac{NSET}{NET}\right) * 100 \tag{1}$$

Regarding the performance of our approach, we noticed, as expected, an increase in execution time when the parameter values are higher. The first plot (Figure 4a) highlights how different values affect the relevance. If the value is high (closer to 100%), it suggests that most entailing triples are useful for goal formulation. This means that KG exploration and entailment filtering are effectively identifying triples that the designer can leverage directly. If the value is low, it signifies that only a few entailing triples are useful in formulating the goals. This might indicate that some triples do not contribute significantly to the high-level goal or are not relevant. The parameters could be adjusted to focus on more relevant triples. Then, the second plot (Figure 4b) highlights how parameter choices affect the final number of leaf goals produced.

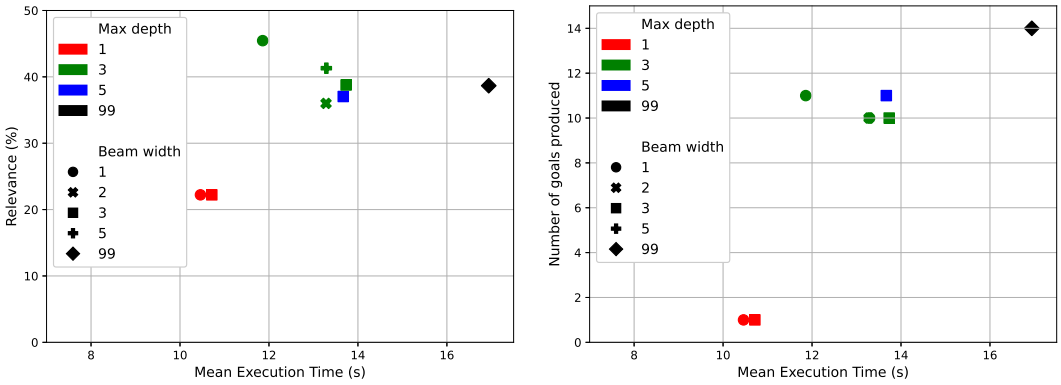
The analysis suggests that the optimal configuration for constructing an initial goal model with a balanced execution time and high relevance is the configuration #2, with a **max depth of 3** and a **beam width of 1**. Using these parameters, we obtain a **Relevance score of 45.454 %**, which is the highest value in our experiment, while **keeping the execution time acceptable** ($T \approx 11.858$ seconds). However, this configuration is not able to help us produce as many goals as the configuration with a **max depth** and **beam width** of 99, therefore inducing an existing loss of relevant information.

The designer can complete each part of the generated goal model by either adding goals from other sources or using our tool with parameter adjustments. This approach ensures that the goal model remains dynamic and adaptable to changing needs.

5 Threats to validity

Concerning the scalability of our approach, the step where we identify anchor points using semantic textual similarity has a complexity which is linear with the number of triples in the graph. While this problem is irrelevant in the context of our evaluation where the KG used contains 119 triples after simple inferences, our method clearly shows its limits in the context of an open KG such as Wikidata[8] or DBpedia[9]. Therefore, our method can only be applied to relatively small, domain-specific KGs. Depending on the context, such graphs could be either

[8] $\sim$ 16 billion triples at the time of writing of this article.
[9] $\sim$ 1.2 billion triples in the English version at the time of writing of this article.



(a) Impact on relevance and time-performance

(b) Impact on the number of produced goals (M10) and time-performance

Fig. 4: Parameter impact on goal model quality and performance (closer to the top-left corner is better)

found as open data, produced for the use case or derived from an open KG by selecting only the triples fulfilling certain criteria (e.g., within a certain distance from a given entity, not linked to irrelevant parts of the topic, etc.).

Additionally, while we conduce a quantitative analysis of the information highlighted by our tool and their relevance, we do not assess the overall quality of the goal models produced. We consider that this issue should be tackled by conducing an experimental evaluation of goal models produced by volunteers using our tool or a baseline alternative. The evaluation could be made based on qualitative criteria (e.g., hierarchical consistency or completeness of the goal model), which would help better characterize the impact of the use of our methodology on goal model creation.

6 Related Work

The automatic or semi-automatic generation of a goal model is a challenging task, and several approaches have been proposed to tackle this problem. Most approaches apply NLP techniques on different text sources.

Güne *et al.* [15] proposed an approach to automatically generate and visualize a goal model from user stories. Heuristics are introduced, designed to extract and organize goals within the structure of a goal model. The evaluation process involved expert feedback, heuristic refinement, and final experiments measuring the similarity between manual and automated models. Case studies in a software development company and academic experiments assessed the tool's usefulness for agile practitioners. The resulting goal models are typically small, focusing on organizing pre-formulated user stories within the goal model. A tool called

ArTu [14] was implemented for automatically generating goal models from user stories.

Das *et al.* [8] proposed an end-to-end framework that uses various NLP techniques to automatically transform a set of unstructured natural language requirement (uNLR) specifications into tGRL goal models. This framework supports goal refinement, associations among goals, resources, and softgoals. The framework follows precise steps to identify goal model artifacts and their relationships across multiple uNLR statements. It was evaluated through case studies on a meeting planning system and an online shopping system. However, the framework may miss some goal model concepts and struggle with identifying relationships between goals and goal decompositions that span multiple statements.

In both approaches, requirements are already provided by stakeholders; the focus is on organizing them in a structured way.

Casagrande *et al.* [6] use NLP to extract goals from the abstracts of research publications related to a specific system category. Sentences that contain *goal-related keywords* are extracted from the abstract. The Stanford parser[10] is used also to generate a syntax tree from sentences. From this syntax tree, a rule-based syntactic pattern is employed to extract a goal as a triplet. Next, the triplet is transformed into a goal representation (structured format). Finally, a goal taxonomy is created by considering the concept of iterative centrality re-ranking, where goals are positioned and linked based on the statistics of their occurrence in the text sources. The generated goal taxonomy serve as a preliminary goal model. However, other sentences that may contain domain-specific information pertinent to the creation of a goal are not considered. Some goals may be missed in the resulting goal model.

Recent research has explored the use of LLMs in the field of RE, offering the potential to drive RE processes [2] and generate goal models.

Nakagawa *et al.* [21] presented an approach for semi-automatically generating a goal model based on a LLM within the MAPE-K loop mechanism, which iteratively refines the goal model. Indeed, MAPE-K loop activities interact with the LLM. The LLM generates the initial goal model, validates it, provides feedback, and integrates new goals based on expert opinions. To achieve this, the authors used prompt engineering techniques to guide the LLM in generating and refining the goal model. The LLM, functioning as a requirements engineer, analyzes the goal model for validation (e.g., refinement links within the goal model) and thereafter provides some expert opinions on the goal model (i.e., feedback to the goal model). Based on these opinions, the LLM, acting again as a requirements engineer, integrates new goals into the goal model. However, the generated goal models could be generic and lack domain-specific precision. This necessitating expert intervention to effectively validate and refine the goal model.

In contrast to existing methods, we introduce an interactive tool designed to assists application designers at each step of creating a goal model rather than validating and adjusting an entire goal model at the end. Our tool effectively

[10] Stanford Parser, a natural language parser tool: `https://nlp.stanford.edu/software/lex-parser.shtml`

leverages knowledge from a domain KG to suggest relevant information that inspires goal creation. Moreover, application designers can add additional goals from other sources to each part of the goal model, enabling a comprehensive and flexible approach to goal modeling. Our tool offers a real-time interaction enables the designer to make adjustments throughout the goal modeling process, resulting in a more precise and contextually accurate goal model.

7 Conclusion

In this paper, we present a tool taking the form of an interface that uses a domain KG to assist application designers create goals and facilitate goal modeling. In comparison with our previous work [1], the approach presented in this paper has been significantly enhanced. We have introduced semantic inference to complement the KG and enhancing the KG exploration algorithm to emphasize both depth limitation and beam search during the entailment exploration process. This improvement is designed to avoid unnecessary and costly explorations, thereby accelerating the overall exploration process. The proposed tool offers flexibility to application designers, allowing them to either freely add goals from other sources or draw inspiration from the suggested information provided by the tool to enrich their goal models. Additionally, to validate our approach, we constructed multiple goal models using our tool with varying parameters. Each model was evaluated on three main criteria: the structure of the goal models, processing efficiency, and quantitative metrics for goal model quality. As future work, we plan to perform an empirical evaluation of our tool to evaluate the degree of facilitation for the designer and the quality of the resulting goal model.

References

1. Abdoul Soukour, S., Aboucaya, W., Georgantas, N.: Leveraging Knowledge Graphs for Goal Model Generation. In: CEUR-WS (ed.) 7th Workshop on Natural Language Processing for Requirements Engineering (NLP4RE). p. 11. Winterthur, Switzerland (Apr 2024)
2. Arora, C., Grundy, J., Abdelrazek, M.: Advancing requirements engineering through generative ai: Assessing the role of llms. In: Generative AI for Effective Software Development, pp. 129–148. Springer (2024)
3. Bowman, S.R., Angeli, G., Potts, C., Manning, C.D.: A large annotated corpus for learning natural language inference. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 632–642. Association for Computational Linguistics, Lisbon, Portugal (Sep 2015)
4. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems **8**, 203–236 (2004)
5. Cailliau, A.: Software requirements engineering: A risk-driven approach. Ph.D. thesis (2018)

6. Casagrande, E., Woldeamlak, S., Woon, W.L., Zeineldin, H.H., Svetinovic, D.: Nlp-kaos for systems goal elicitation: Smart metering system case study. IEEE Transactions on Software Engineering **40**(10), 941–956 (2014)
7. Castro Ferreira, T., Gardent, C., Ilinykh, N., van der Lee, C., Mille, S., Moussallem, D., Shimorina, A.: The 2020 bilingual, bi-directional WebNLG+ shared task: Overview and evaluation results (WebNLG+ 2020). In: Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+). pp. 55–76. Association for Computational Linguistics, Dublin, Ireland (Virtual) (12 2020), https://aclanthology.org/2020.webnlg-1.7
8. Das, S., Deb, N., Cortesi, A., Chaki, N.: Extracting goal models from natural language requirement specifications. Journal of Systems and Software **211** (2024)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019)
10. Espada, P., Goulão, M., Araújo, J.: Measuring complexity and completeness of kaos goal models. In: Workshop on Empirical Requirements Engineering (EmpiRE 2011). pp. 29–32. IEEE (2011)
11. Espada, P., Goulão, M., Araújo, J.: A framework to evaluate complexity and completeness of kaos goal models. In: Advanced Information Systems Engineering: 25th International Conference, CAiSE 2013, Valencia, Spain, June 17-21, 2013. Proceedings 25. pp. 562–577. Springer (2013)
12. Franch, X., López, L., Cares, C., Colomer, D.: The i* Framework for Goal-Oriented Modeling, pp. 485–506. Springer International Publishing (07 2016)
13. Gralha, C., Araújo, J., Goulão, M.: Metrics for measuring complexity and completeness for social goal models. Information Systems **53**, 346–362 (2015)
14. Günes, T., Öz, C.A., Aydemir, F.B.: Artu: a tool for generating goal models from user stories. In: 2021 IEEE 29th International Requirements Engineering Conference (RE). pp. 436–437. IEEE (2021)
15. Güne, T., Aydemir, F.B.: Automated goal model extraction from user stories using nlp. In: 2020 IEEE 28th International Requirements Engineering Conference (RE). pp. 382–387 (2020). https://doi.org/10.1109/RE48521.2020.00052
16. Ke, P., Ji, H., Ran, Y., Cui, X., Wang, L., Song, L., Zhu, X., Huang, M.: JointGT: Graph-text joint representation learning for text generation from knowledge graphs. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. Association for Computational Linguistics, Online (Aug 2021)
17. van Lamsweerde, A.: Goal-oriented requirements engineering: a guided tour. In: Proceedings Fifth IEEE International Symposium on Requirements Engineering. pp. 249–262 (2001). https://doi.org/10.1109/ISRE.2001.948567
18. van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley Publishing, 1st edn. (2009)
19. Letier, E.: Reasoning about agents in goal-oriented requirements engineering (2002)
20. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
21. Nakagawa, H., Honiden, S.: Mape-k loop-based goal model generation using generative ai. In: 2023 IEEE 31st International Requirements Engineering Conference Workshops (REW). pp. 247–251. IEEE (2023)

22. Nan, L., Radev, D., Zhang, R., Rau, A., Sivaprasad, A., Hsieh, C., Tang, X., Vyas, A., Verma, N., Krishna, P., Liu, Y., Irwanto, N., Pan, J., Rahman, F., Zaidi, A., Mutuma, M., Tarabar, Y., Gupta, A., Yu, T., Tan, Y.C., Lin, X.V., Xiong, C., Socher, R., Rajani, N.F.: DART: Open-domain structured data record to text generation. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 432–447. Association for Computational Linguistics (Jun 2021)
23. Nie, Y., Chen, H., Bansal, M.: Combining fact extraction and verification with neural semantic matching networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 6859–6866 (Jul 2019)
24. Nie, Y., Williams, A., Dinan, E., Bansal, M., Weston, J., Kiela, D.: Adversarial NLI: A new benchmark for natural language understanding. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 4885–4901. Association for Computational Linguistics, Online (Jul 2020)
25. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: A method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. p. 311318. ACL '02, Association for Computational Linguistics, USA (2002)
26. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research **21**(140), 1–67 (2020)
27. Rebuffel, C., Scialom, T., Soulier, L., Piwowarski, B., Lamprier, S., Staiano, J., Scoutheeten, G., Gallinari, P.: Data-QuestEval: A Reference-less Metric for Data-to-Text Semantic Evaluation. In: 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 8029–8036. Association for Computational Linguistics, Punta Cana, Dominican Republic (Nov 2021)
28. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (Nov 2019)
29. Ribeiro, L.F.R., Schmitt, M., Schütze, H., Gurevych, I.: Investigating pretrained language models for graph-to-text generation. In: Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI. pp. 211–227. Association for Computational Linguistics, Online (Nov 2021)
30. Schlegel, V., Nenadic, G., Batista-Navarro, R.: Beyond leaderboards: A survey of methods for revealing weaknesses in natural language inference data and models. arXiv preprint arXiv:2005.14709 (2020)
31. Shimada, H., Nakagawa, H., Tsuchiya, T.: Goal model construction based on user review classification. In: REFSQ Workshops (2019)
32. Williams, A., Nangia, N., Bowman, S.: A broad-coverage challenge corpus for sentence understanding through inference. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 1112–1122. Association for Computational Linguistics (2018)
33. Zhang, K., Balalau, O., Manolescu, I.: FactSpotter: Evaluating the Factual Faithfulness of Graph-to-Text Generation. In: Findings of EMNLP 2023 - Conference on Empirical Methods in Natural Language Processing. Singapore, Singapore (Dec 2023), `https://hal.science/hal-04257838`
34. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: Bertscore: Evaluating text generation with bert (2020)