Leveraging Knowledge Graphs for Goal Model Generation

Shahin Abdoul Soukour[1,*], William Aboucaya[1] and Nikolaos Georgantas[1]

[1]*Inria, Paris, France*

Abstract

KAOS is one of the most widely used Goal-Oriented Requirements Engineering (GORE) methods. The goal model is the central element of KAOS, employed to represent the goals of a system in the form of a hierarchy, where higher-level goals are refined into lower-level ones. The process of constructing a KAOS goal model for a new application can present challenges, requiring significant time and effort. Existing approaches have tried to partially automate the construction of goal models, however, this largely remains a complex, manual task. In this paper, we propose leveraging domain knowledge in the form of a Knowledge Graph (KG), which can assist the application designer in creating goals that are inspired from this knowledge. To accomplish this, we leverage semantic similarity measurement and Natural Language Inference (NLI) to effectively extract triples from the KG that are relevant to a high-level goal formulated by the designer. The extracted triples are further processed through sentiment analysis and graph-to-text generation, before presented to the designer. Via step-by-step interaction with our solution, the designer can gradually refine their initial goals into a goal hierarchy. We demonstrate our approach by applying it to the design of a flood management system, based on a handcrafted domain KG.

1. Introduction

Goal-Oriented Requirements Engineering (GORE) is a specific approach to requirements engineering, which focuses on capturing and modeling the goals that stakeholders want to achieve with a software system. Several goal-oriented modeling methods have been proposed [1, 2, 3, 4]. One of the most popular and recognized methods, both in academia and industry, is "*Keep All Objectives Satisfied*", more commonly called **KAOS** [1]. It provides a representation of goals in an explicit way (**goal model**), where high-level goals (strategic, global) are refined (or decomposed) into lower-level ones (operational, local, design-specific) to achieve a comprehensive understanding of the system. Ultimately, the leaf goals of the goal model are elementary goals that should be mapped to operations of specific software components.

Creating a KAOS goal model for a new application is a complex and challenging process, requiring a substantial investment of time and effort from the application designer. Complexity comes from both identifying the goals of the new system and organizing them into a goal model.

In the literature, a few approaches have proposed to generate automatically or semi-automatically a goal model by extracting goals from a small text corpus. Güneş *et al.* [5] rely on user stories as text sources. 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 sentences containing *goal-related keywords*; from these goals a goal taxonomy is created. However, other sentences that may contain relevant domain-specific information for creating a goal are not taken into account. Both approaches attempt to build a complete goal model from the reference text sources. The application

In: D. Mendez, A. Moreira, J. Horkoff, T. Weyer, M. Daneva, M. Unterkalmsteiner, S. Bühne, J. Hehn, B. Penzenstadler, N. Condori-Fernández, O. Dieste, R. Guizzardi, K. M. Habibullah, A. Perini, A. Susi, S. Abualhaija, C. Arora, D. Dell'Anna, A. Ferrari, S. Ghanavati, F. Dalpiaz, J. Steghöfer, A. Rachmann, J. Gulden, A. Müller, M. Beck, D. Birkmeier, A. Herrmann, P. Mennig, K. Schneider. Joint Proceedings of REFSQ-2024 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. Co-located with REFSQ 2024. Winterthur, Switzerland, April 8, 2024.

*Corresponding author.

abdoul-shahin.abdoul-soukour@inria.fr (S. Abdoul Soukour); william.aboucaya@inria.fr (W. Aboucaya); nikolaos.georgantas@inria.fr (N. Georgantas)
https://williamaboucaya.github.io/ (W. Aboucaya); https://mimove.inria.fr/members/nikolaos-georgantas/ (N. Georgantas)
0009-0005-5026-712X (S. Abdoul Soukour); 0000-0003-2413-6968 (W. Aboucaya); 0000-0001-5704-4889 (N. Georgantas)

designer can intervene only at the end of the process to verify the validity of the model and use or modify (parts of) it.

In contrast to these approaches, our solution in this paper aims at offering interactive assistance to the application designer for gradually conceiving and refining their goal model. We provide inspiration to the designer based on existing domain knowledge. In particular, we propose leveraging domain knowledge in the form of a **Knowledge Graph** (KG). Several online open KGs, such as DBpedia [7] and Wikidata [8], have been created based on Semantic Web technologies and the Linked Open Data (LOD) paradigm. These KGs are very big and typically address the general public; nevertheless, extracting relevant information from them for a specific domain is worth investigating. On the other hand, domain-specific KGs are harder to find, particularly those that encompass high-level concepts or goals describing a system domain (e.g., traffic management in a city, treatment plans in healthcare, etc.). Typically, information coming from generalist or domain-specific KGs expresses facts. On the other hand, a goal describes a target system condition that should be achieved (or one that should be avoided) [1]. In our approach, facts from a KG become or inspire target conditions related to goals. For the purposes of this paper and to support our use case based validation, we have elaborated a compact domain-specific KG related to flood warning.

To effectively exploit domain knowledge, we propose an approach for exploring a KG that relies on several **Natural Language Processing** (NLP) techniques based on pre-trained language models. Our graph exploration approach includes:

- Measuring **semantic textual similarity** between a high-level goal formulated by the application designer and the triples[1] of the KG in order to identify **anchor triples** inside the KG. Anchor triples and their neighbor triples denote KG areas that can most probably provide relevant information to the application designer.
- Employing **Natural Language Inference** (NLI) to identify a **textual entailment** relation between the high-level goal and one or several combined triples. If such entailment holds, these triples can be used to refine this goal. In particular, we propose to use textual entailment to approximate the *contribution* relation between a goal and one of its subgoals in which this goal is refined: satisfaction of the subgoal contributes to satisfaction of the parent goal.
- Using **Sentiment Analysis** to identify positive and negative connotations in triples. Triples with negative connotations describe facts that should be avoided. By stating explicitly this connotation, we improve entailment detection between such triples and a formulated goal. Additionally, based on this analysis, triples extracted from the KG are presented to the application designer with the qualification [ACHIEVE] or [AVOID], which can facilitate the creation of related goals.
- Applying **Graph-to-Text** (G2T) generation to build grammatically correct, fluid and semantically coherent text from one or more triples extracted from the KG before presenting them to the application designer. G2T simplifies the comprehension of the entailed triples and facilitates the inspiration of new goals.

Based on these techniques and via step-by-step interaction with our solution, the designer can gradually refine their initial goals into a goal hierarchy that encompasses all the relevant information extracted from the KG. Certainly, this information can and should be complemented with information coming from other sources (inputs from stakeholders, designer's expertise) to generate a complete goal model. To validate our approach, we demonstrate its functioning by applying it to the design of a flood management system, based on our flood warning use case and domain KG. Besides inspecting the relevance of the generated goal model, we measure the amount of relevant information that was successfully extracted from the KG as well as the time-related performance of our solution.

The rest of the paper is structured as follows: Section 2 presents some background notions related to NLI and G2T generation. Section 3 introduces our method for building a KAOS goal model. Section 4

[1]A **RDF triple**, or a semantic triple, is a representation of data from a knowledge graph composed of a **subject** and an **object** nodes linked by a directed edge from the subject to the object called the **predicate**.

presents the demonstration of our approach. Section 5 mentions some threats to the validity of the proposed approach. Section 6 discusses related works. Finally, in Section 7, we conclude and give directions for future work.

The code used in the production of this paper is available at https://github.com/ShahinAbdoulSoukour/KG_for_goal_model_generation.

2. Background

2.1. Natural Language Inference

Natural Language Inference (NLI), also referred as **Textual Entailment Recognition** (TER), is a task aiming at labelling a pair of natural language texts, the premise and the hypothesis, based on the three following cases: the hypothesis can be deduced from the premise (**entailment**); the negation of the hypothesis can be deduced from the premise (**contradiction**); neither the hypothesis nor its negation can be deduced from the premise (**neutral**).

Multiple approaches have been proposed to tackle NLI tasks, but here we will only focus on the use of language models trained specifically for this task. This approach is generally performed using Transformers-based models, such as BERT [9] or RoBERTa [10], since these models can leverage their advanced language "understanding" for specific tasks including NLI, outperforming other approaches. Multiple datasets have been proposed for the training and assessment of models on this task, such as the Stanford NLI (SNLI) [11], Multi-genre NLI (MNLI) [12] or Adversarial NLI (ANLI) [13] corpuses. In this paper, NLI tasks are performed using a RoBERTa$_{LARGE}$ model trained using the SNLI, MNLI, ANLI and FEVER-NLI [14] corpuses[2]. Performances of this model on the different datasets used for training are available in [13].

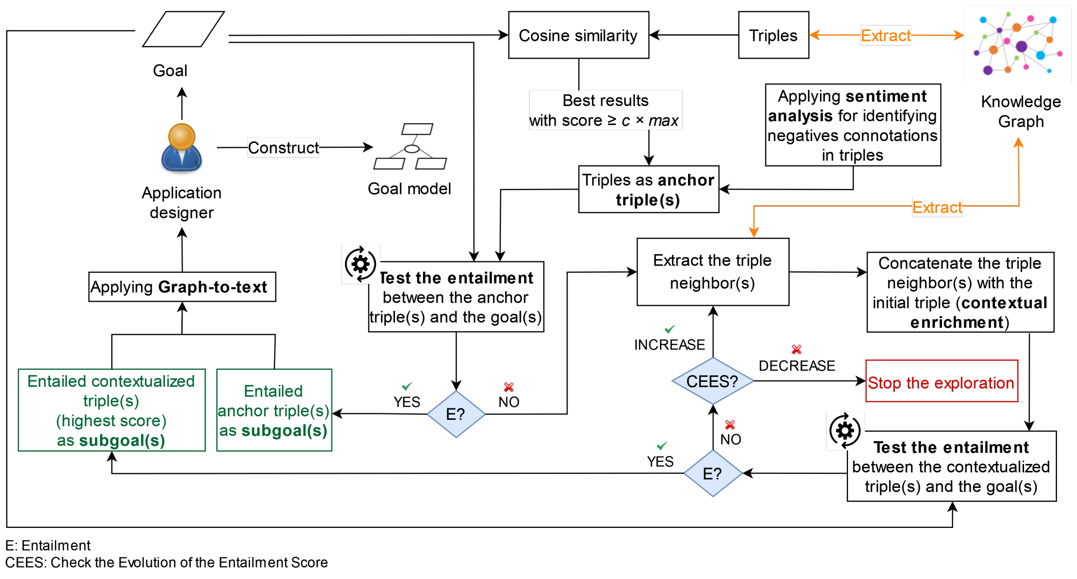
2.2. Graph-to-text generation

Graph-to-Text generation (G2T) is a task aiming at rendering the entities in a graph and their relations as a natural language text. The objective is generally to make the information stored in a KG readable for non-specialist users. The most common criterion of evaluation of G2T models is the similarity of the text generated with a reference description of the graph written by a human, using metrics such as BLEU [15] or BERTScore [16]. However, more specific metrics for G2T tasks have been proposed to assess the *factual faithfulness* of the text generated to the initial data, such as Data-QuestEval [17] or FactSpotter [18]. Currently, most state-of-the-art models for G2T are based on a Transformers architecture [19, 20]. The pre-training and evaluation of these models is generally performed using datasets containing pairs composed of a graph and a text summarizing its content, such as DART [21] or WebNLG [22]. In this paper, G2T tasks are performed using a T5$_{BASE}$ [23] model trained using the WebNLG'20 dataset[3]. Performances of this model on multiple metrics, including the factual faithfulness of the generated texts in relation to the original data, are available in [18].

3. Method

We elaborate in this section our method for assisting the application designer in building a KAOS goal model for a new application by relying on knowledge coming from a pre-existing domain KG. The designer launches the method (which takes the form of an interactive tool) by formulating an initial goal that characterizes the new application, typically a high-level one. After exploring the KG, the tool returns one or more pieces of text that analyze or refine the submitted goal. By using directly these texts or getting inspired from them, the designer can create one or more subgoals for the initial goal. Our approach comprises the following steps (see Figure 1).

[2]Model available at https://huggingface.co/ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli
[3]Model available at https://huggingface.co/Inria-CEDAR/WebNLG20T5B



Goal
Cosine similarity
Triples
Extract
Knowledge Graph
Best results with score ≥ c × max
Applying sentiment analysis for identifying negatives connotations in triples
Application designer
Construct
Goal model
Triples as anchor triple(s)
Extract
Test the entailment between the anchor triple(s) and the goal(s)
Extract the triple neighbor(s)
Concatenate the triple neighbor(s) with the initial triple (contextual enrichment)
Applying Graph-to-text
INCREASE
DECREASE
CEES?
Stop the exploration
Entailed contextualized triple(s) (highest score) as subgoal(s)
Entailed anchor triple(s) as subgoal(s)
YES
E?
NO
NO
YES
E?
Test the entailment between the contextualized triple(s) and the goal(s)
E: Entailment
CEES: Check the Evolution of the Entailment Score

Figure 1: Steps of our approach to facilitate goal models generation using a knowledge graph.

1. **Formulate a goal.** The application designer formulates a goal. Ideally, this goal should be quite abstract and not composite. A composite goal should be split into more elementary goals. This allows more efficient exploration of the KG based on entailment. In particular, our experimentation with the employed NLI model showed that in most cases an entailment is correctly identified when the hypothesis is an abstraction of the premise.

2. **Search for anchor triples in the KG.** Based on the goal submitted by the application designer, relevant triples are sought in the KG. Querying a KG in order to find elements of interest can be tedious and time consuming. It requires knowledge of the query language (typically SPARQL) and preferably some understanding of the concepts and structure of the KG. If the latter does not hold, SPARQL queries return empty results and need to be reformulated. Even when trying keyword search, a SPARQL query will not return any results, if an exact match is not found. Furthermore, while the use of a visual representation can facilitate information retrieval for small-scale KGs, this solution tends to be less effective at a larger scale. To tackle this issue, we use SBERT [24] to compute the semantic textual similarity between each triple of the KG and the expressed goal. Using semantic textual similarity rather than simpler methods like TF-IDF or BLEU [15] enables our method to identify texts evoking similar topics without relying on the existence of common words. This means that our method takes into account the semantic meaning and context, facilitating a more precise identification of related texts. 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]. These triples are regarded as **anchor triples** in the KG. The main advantage of using such a threshold for similarity is to improve the flexibility of the number of anchor triples selected, as opposed to selecting a fixed number N of anchor triples with the highest similarity values. For instance, opting for a high number of anchor triples may be relevant in scenarios where the goals encompass a wide range of entities in the KG. On the other hand, for more specific goals, it may be more appropriate to associate only a few anchor triples. This flexibility ensures that the selection of anchor triples aligns with the desired scope, thus improving the efficiency of the overall process.

3. **Identify connotation in anchor triples by applying sentiment analysis.** Triples are distinguished based on their connotation through sentiment analysis[5] (a process of tagging data according

[4]Higher values of c improves the quality of the retrieved triples while reducing their number.
[5]The model used for the demonstration of our method is available at https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest. After an exhaustive evaluation of the results produced by this model on the triples of our example KG, we obtained a F1-score of 1 (perfect annotations on the whole dataset).

to their predicted sentiment: positive, negative or neutral). Triples with a positive connotation describe facts that we want to achieve, while triples with a negative connotation describe facts that we want to avoid. When testing entailment between negatively connotated triples and a submitted goal, we prefix each one with "*Prevent that*". This improves entailment detection.

4. **Test entailment between anchor triples and the goal.** NLI is employed to evaluate entailment between each anchor triple as premise and the submitted goal as hypothesis. If entailment holds for an anchor triple, the fact expressed by the triple can be considered as contributing to the goal. In other words, this triple can be the basis for formulating a subgoal of the initial goal. The entailing anchor triples are kept.
5. **Add context to non-entailing anchor triples.** If entailment does not hold between an anchor triple and the submitted goal, this may be due to a semantic gap that NLI cannot bridge, even if the fact expressed by the triple represents a potential contribution to the goal. To avoid missing these cases, we try to bridge this gap by contextually enriching such triples. We look for potentially relevant context among the neighbor triples of an anchor triple, i.e., triples that share a concept (subject or object) with the anchor triple. We extract neighbor triples by using SPARQL, while excluding neighbor triples that were found entailing in the previous step. An anchor triple and a neighbor triple are concatenated into a contextualized triple.
6. **Test entailment between contextualized triples and the goal.** NLI is employed again to evaluate entailment between a contextualized triple and the submitted goal. Among all the contextualized triples resulting from an anchor triple, we retain the one with the highest entailment score. If additionally entailment holds, we keep this triple. If entailment does not hold, we repeat Steps 5 and 6. In this case, we add one more context triple to the already contextualized triple by choosing among its neighbor set (which is now extended as the contextualized triple comprises two triples). We apply this process as long as the entailment score increases upon each new context addition. We stop if we reach an entailment or if the entailment score decreases.
7. **Return entailing triples and create subgoals.** Both anchor triples and contextualized triples that were found as entailing are successful outputs of the tool. Before presenting them to the application designer, graph-to-text generation is employed to transform them into natural language texts. Additionally, based on the result of the sentiment analysis in Step 3, the generated pieces of text are annotated with the corresponding semantics. The prefix [ACHIEVE] is added to texts resulting from anchor triples with a positive or neutral connotation, while the prefix [AVOID] is added to texts resulting from anchor triples with a negative connotation. Accordingly, the application designer can create one or more ACHIEVE or AVOID[6] subgoals for the initial goal, by relying directly on the texts returned by the tool or modifying them as they wish.
8. **Repeat the process.** This interaction can be repeated several times. Each time the designer submits to the tool a goal, which can be a subgoal created from a previous iteration or a completely new goal. When developing a specific branch in the goal hierarchy, the tool maintains a history of the triples that have already been used by the application designer to create goals. Based on this history, a triple cannot be used as an anchor triple to create subgoals for a goal if it has been used to produce that goal or one of its parents. This rule aims at avoiding redundancy and facilitating the exploration of new concepts in the KG. The refinement process concerning a specific branch in the goal hierarchy ends when the tool returns no new information in response to a designer's request.

4. Demonstration of the approach on a use case

To demonstrate the approach proposed in this paper, we apply it to the creation of a goal model for a flood management system. We rely on the implementation of our method as a software tool (coded in Python in the form of a Jupyter notebook) as well as on a domain KG related to flood warning that

[6]These are goal types identified in KAOS [1].

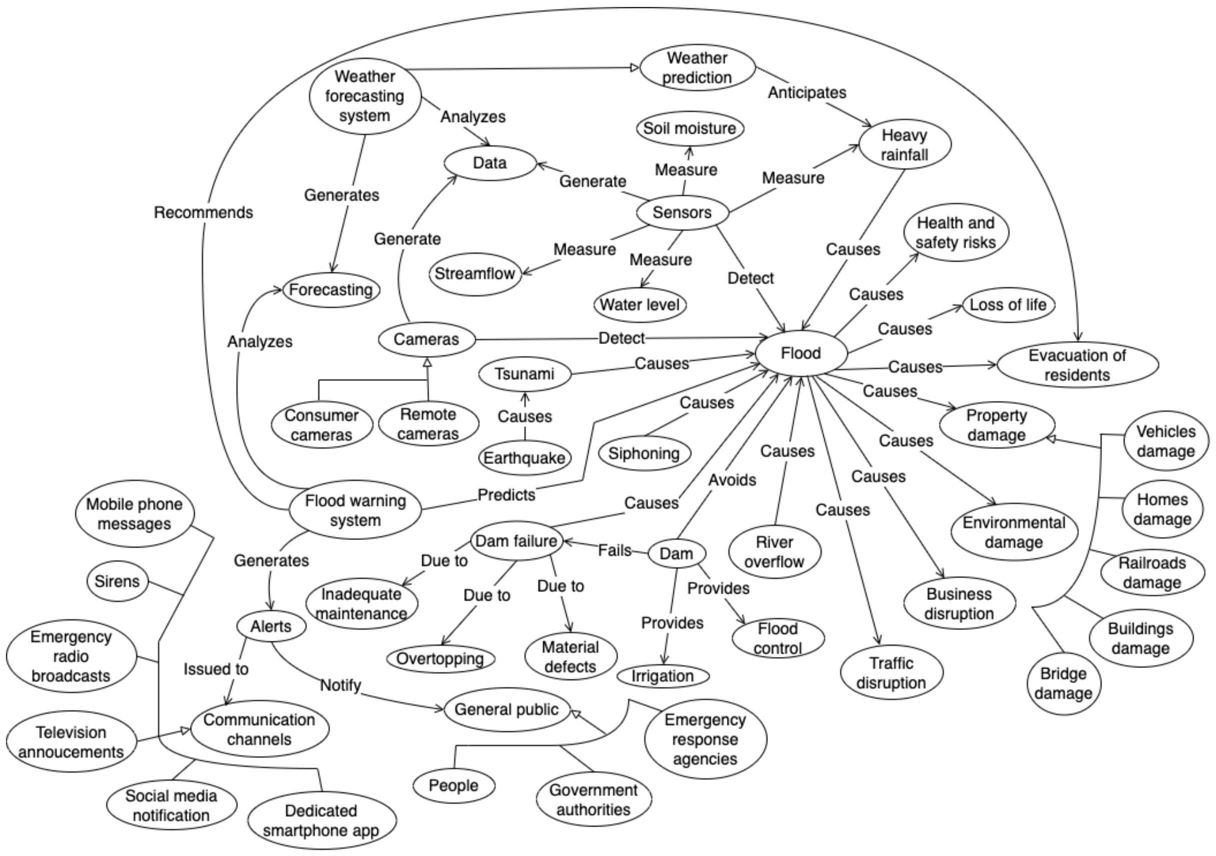


Figure 2: Domain-specific knowledge graph focused on flood detection and management

we have created to support our demonstration (see Figure 2). The KG is composed of 49 vertices and 54 edges describing the fundamentals of flood management – the potential causes of floods, how to detect them, the expected impacts of a flood on people and properties, and how to prevent or mitigate these impacts. We use a configuration of our tool with a confidence parameter $c = 0.85$ for semantic similarity. We observed that the resulting threshold value limits the risk of identifying irrelevant triples as anchor triples while returning a high enough number of triples. We demonstrate the functioning of the tool from the point of view of a designer as follows:

a. We produce an initial high-level goal as abstract as possible. In this example, the high-level goal is "*Anticipate the impact of floods on people*".

b. The goal is given as an input to our tool, which returns a series of natural language texts created from entailing triples retrieved from the KG.

c. Based on the output of the tool, we propose one or multiple texts to refine our initial goal. For demonstration purposes, we do not add any information in the goals outside of that returned by our tool.

d. We repeat Steps b and c for each subgoal produced until the tool stops returning new information or the information returned does not allow us to refine a goal.

The resulting goal model is displayed in Figure 3. Our tool highlights a total of 19 triples, i.e., 35.2 % of the 54 triples contained in our graph. Out of these 19 triples, 14 of them (73.7 %) helped us refine the goal(s) they were associated with. Moreover, the two main groups of irrelevant triples – inheritance relationships of "*Property damage*" and descriptions of elements related to the "*Dam*" node – were not highlighted by our tool. We therefore consider that our tool did facilitate the generation of our goal model. However, certain groups of relevant triples – inheritance relationships of the "*Communication channels*" and "*General public*" entities – were not highlighted by our tool. This leads us to consider that

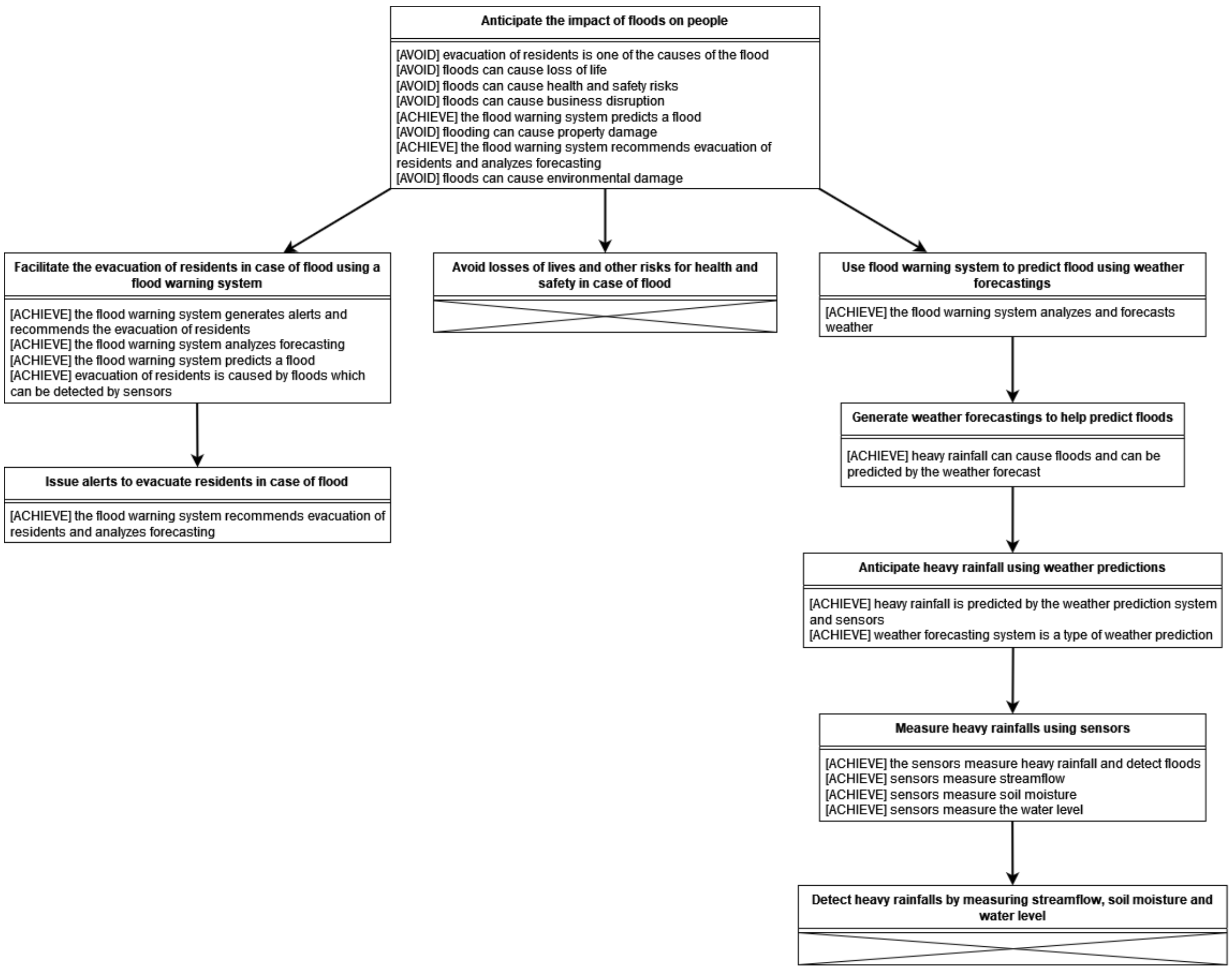

Anticipate the impact of floods on people
[AVOID] evacuation of residents is one of the causes of the flood
[AVOID] floods can cause loss of life
[AVOID] floods can cause health and safety risks
[AVOID] floods can cause business disruption
[ACHIEVE] the flood warning system predicts a flood
[AVOID] flooding can cause property damage
[ACHIEVE] the flood warning system recommends evacuation of residents and analyzes forecasting
[AVOID] floods can cause environmental damage
Facilitate the evacuation of residents in case of flood using a flood warning system
[ACHIEVE] the flood warning system generates alerts and recommends the evacuation of residents
[ACHIEVE] the flood warning system analyzes forecasting
[ACHIEVE] the flood warning system predicts a flood
[ACHIEVE] evacuation of residents is caused by floods which can be detected by sensors
Issue alerts to evacuate residents in case of flood
[ACHIEVE] the flood warning system recommends evacuation of residents and analyzes forecasting
Avoid losses of lives and other risks for health and safety in case of flood
Use flood warning system to predict flood using weather forecastings
[ACHIEVE] the flood warning system analyzes and forecasts weather
Generate weather forecastings to help predict floods
[ACHIEVE] heavy rainfall can cause floods and can be predicted by the weather forecast
Anticipate heavy rainfall using weather predictions
[ACHIEVE] heavy rainfall is predicted by the weather prediction system and sensors
[ACHIEVE] weather forecasting system is a type of weather prediction
Measure heavy rainfalls using sensors
[ACHIEVE] the sensors measure heavy rainfall and detect floods
[ACHIEVE] sensors measure streamflow
[ACHIEVE] sensors measure soil moisture
[ACHIEVE] sensors measure the water level
Detect heavy rainfalls by measuring streamflow, soil moisture and water level

Figure 3: Goal model produced using our tool. Each box contains the goal followed by the texts generated for it.

addition of inference rules for inheritance relationships could help identifying the relevance of such triples[7]. Implementation of inference rules in our tool for common predicates (e.g., *rdf:type*, *owl:sameAs*) is part of our future work toward the facilitation of goal model generation using KGs.

To further characterize these results, we present the outputs produced by our tool in the different steps of our method (as described in Section 3) for the goal "*Use flood warning system to predict flood using weather forecastings*". In Step 2 of our method, we search for anchor triples in the KG. We exclude the triples "*Flood warning system Predicts Flood*" and "*Flood warning system Analyzes Forecasting*" as they were used to create this goal.
We identify the following anchor triples:

(i) *Flood warning system Generates Alerts* ($similarity = 0.68$)

(ii) *Flood warning system Recommends Evacuation of residents* ($similarity = 0.67$)

(iii) *Sensors Detect Flood* ($similarity = 0.64$)

(iv) *Weather prediction Anticipates Heavy rainfall* ($similarity = 0.60$)

(v) *Weather forecasting system is a type of Weather prediction* ($similarity = 0.59$)

(vi) *Weather forecasting system Generates Forecasting* ($similarity = 0.59$)

[7]For example, in the context of this graph, the rule "IF *subject predicate parent* AND *child rdf:type parent* THEN *subject predicate child*" could help us produce a more complete graph.

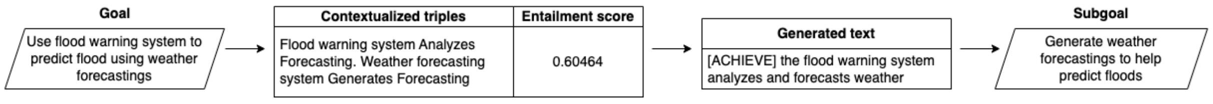

Goal
Use flood warning system to predict flood using weather forecastings
Contextualized triples
Entailment score
Flood warning system Analyzes Forecasting. Weather forecasting system Generates Forecasting
0.60464
Generated text
[ACHIEVE] the flood warning system analyzes and forecasts weather
Subgoal
Generate weather forecastings to help predict floods

Figure 4: Subgoal creation from the goal ”*Use flood warning system to predict flood using weather forecastings*”

Steps 3 and 4 inform us that each of these anchor triples is non-entailing with respect to the goal. Consequently, we explore their neighbors to identify potentially relevant elements of context. This graph exploration performed in Steps 5 and 6 does not lead us to identify relevant context for triples *(i)* to *(v)*. However, pairing triple *(vi)* with its neighbor "*Flood warning system Predicts Flood*" allows us to produce an entailing contextualized triple. In Step 7, using G2T generation, we transform triple *(vi)* and its relevant context into a natural language version: "*[ACHIEVE] the flood warning system analyzes and forecasts weather*". A subgoal can then be generated by the designer (see Figure 4).

Finally, execution time is a key metric for our interactive tool. For the goal "*Anticipate the impact of floods on people*", our tool outputs the entailing triples in 7.68 seconds on average. Additionally, text generation from these triples is performed in 2.49 seconds per text on average. These results are based on 5 executions of our code using an Nvidia 3080 RTX GPU with 32 GB RAM.

5. Threats to validity

For validating our approach, we produced our own KG. While we consider that our graph accurately represents the topic that we tackled as an example – flood prevention and mitigation at the scale of city –, we should acknowledge that it has been produced in the context of our research and therefore may lack certain elements which could have been included by specialists of the domain. Our approach leverages the information stored in a KG and therefore the quality of the texts produced and consequently of the resulting goal model relies highly on the correctness and completeness of the KG.

On a different note, the complexity of our approach is linear with the number of triples in the graph. While this is perfectly acceptable for relatively small graphs, it is not possible to use our tool as it is currently implemented on open KGs such as DBpedia or private KGs containing thousands of triples without greatly increasing the execution time. Consequently, we need to limit the number of triples analyzed in our approach for large KGs, for example by using a subgraph containing only the nodes within a certain distance from an entity representing a key concept for the goal model to create.

6. Related work

Generating a goal model in an automatic or semi-automatic manner is a challenging task, and various approaches have addressed this issue. Typically, these approaches apply NLP techniques on different text sources (e.g., user stories [5], abstracts of research publications [6], user reviews [25]).

Güneş *et al.* [5] proposed an approach to automatically generate and visualize a goal model from a set of user stories. Several heuristics are introduced to elicit goals and to organize them in a goal model. The produced goal model is typically small given the size of user stories. Since requirements are already formulated in the user stories, this work focuses on their structuring in the goal model.

Casagrande *et al.* [6] use NLP to extract goals from the abstracts of research publications related to a specific system category by looking for sentences that contain *goal-related keywords*. The Stanford parser[8] is used to generate a syntax tree from sentences. Then, a rule-based syntactic pattern is used to extract a goal from a syntax tree as a triplet. Finally, a goal taxonomy is created by taking into account the concept of iterative centrality re-ranking (goals are positioned and linked based on the statistics of their occurrence in the text sources). The generated goal taxonomy represents a preliminary goal

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

model. However, other sentences that may contain pertinent domain-specific information for creating a goal are not taken into account. Certain goals may be missed in the resulting goal model.

Shimada *et al.* [25] proposed a systematic approach to generate a goal model from user reviews of an application. The authors use a hierarchical clustering method, which couples the clusters by order of distance between them and organizes them in a tree structure (reviews clustering). Clusters are interpreted as goals in the goal model. In the next step, each cluster is labeled with a goal description. In particular, words in a cluster are weighted and several words that characterize the reviews in the cluster are selected as a goal description. Similarly to Güneş *et al.* [5], requirements are already formulated here in the user reviews.

Nakagawa *et al.* [26] proposed a method to extract goals from requirements documents and to generate a goal model in a semi-automatic way. An analyst needs to answer a flow of questions (pre-defined) based on the requirement documents, which are asked interactively by the tool developed by the authors. The goal model is constructed based on the answers. The tool decides the location of goals in the goal model. This structuring of the goal model can be interesting. However, the goals need to be already provided in requirements documents.

Watanabe *et al.* [27] presented an open-source editing tool for automatically generating an initial KAOS goal model from a requirements description. The automatic construction of a goal model consists of 3 steps: parsing sentences into words, extraction of important sentences identified based on specific keywords, and extraction and linking goals based on word-to-word dependencies.

Research has also aimed at completing an existing goal model. Shibaoka *et al.* [28] used a domain ontology as domain knowledge in order to identify missing goals. The approach maps certain elements of the goal model in-progress onto the domain ontology. After that, the ontology inference mechanism deduces related ontological concepts that should possibly be added to the goal model. The suggested concepts are then added to the goal model by the requirement analyst. The goal model is thus enriched with new subgoals. However, this approach cannot be used for building an entire new goal model.

Compared to these different approaches, we propose an interactive method that assists the application designer at every stage of producing a goal model. In addition, while most works rely on requirements documents to build a goal model, our approach effectively exploits knowledge from a domain KG for suggesting or inspiring goals.

7. Conclusion

This research focuses on the generation of goal models using domain-specific KGs. The creation of goal models can be facilitated by highlighting relevant elements of knowledge from the KGs to refine high-level goals. We first propose a method for identifying relevant triples in order to refine a given high-level goal by leveraging NLI and sentiment analysis. Then, we demonstrate our method on a particular use case using a KG that describes the fundamentals of flood management. Our demonstration shows that our tool highlights pertinent elements of knowledge to help designers produce goal models. However, a more systematic evaluation of the approach is needed. Therefore, we propose to perform an empirical evaluation comparing the goal models produced using our method as well as other tools.

Toward the elaboration of a framework for assisted goal model creation using KGs, we propose several leads for future work. First, the identification of triples referenced in a goal is currently done by the designer. This work could be automated using factual faithfulness evaluation models such as FactSpotter [18] to assess the presence of certain triples in the goal's text. Second, we propose to implement inference rules in our tool to improve the understanding of semantic relationships in the graph rather than relying only on the analysis of natural language transcriptions of the triples. Third, we aim at improving the scalability of our tool with regard to the number of triples in the graph to allow its use on open KGs in contexts where no domain-specific KG is available. This future work would need the automated creation of subgraphs from open KGs to identify only relevant information for the domain of the goal model.

References

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

[2] E. Letier, Reasoning about agents in goal-oriented requirements engineering, 2002.

[3] E. Yu, Towards modelling and reasoning support for early-phase requirements engineering, in: Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering, 1997, pp. 226–235. doi:10.1109/ISRE.1997.566873.

[4] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, Tropos: An agent-oriented software development methodology, Autonomous Agents and Multi-Agent Systems 8 (2004) 203–236. doi:10.1023/B:AGNT.0000018806.20944.ef.

[5] T. Güneş, F. B. Aydemir, Automated goal model extraction from user stories using nlp, in: 2020 IEEE 28th International Requirements Engineering Conference (RE), 2020, pp. 382–387. doi:10.1109/RE48521.2020.00052.

[6] E. Casagrande, S. Woldeamlak, W. L. Woon, H. H. Zeineldin, D. Svetinovic, Nlp-kaos for systems goal elicitation: Smart metering system case study, IEEE Transactions on Software Engineering 40 (2014) 941–956. doi:10.1109/TSE.2014.2339811.

[7] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: The Semantic Web, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 722–735. doi:10.1007/978-3-540-76298-0_52.

[8] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez, D. Vrandečić, Introducing wikidata to the linked data web, in: The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I 13, Springer, 2014, pp. 50–65.

[9] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, 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, 2019, pp. 4171–4186. URL: https://aclanthology.org/N19-1423. doi:10.18653/v1/N19-1423.

[10] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).

[11] S. R. Bowman, G. Angeli, C. Potts, C. D. Manning, A large annotated corpus for learning natural language inference, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 632–642. URL: https://aclanthology.org/D15-1075. doi:10.18653/v1/D15-1075.

[12] A. Williams, N. Nangia, S. Bowman, 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), Association for Computational Linguistics, 2018, pp. 1112–1122.

[13] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, D. Kiela, Adversarial NLI: A new benchmark for natural language understanding, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 4885–4901. URL: https://aclanthology.org/2020.acl-main.441. doi:10.18653/v1/2020.acl-main.441.

[14] Y. Nie, H. Chen, M. Bansal, Combining fact extraction and verification with neural semantic matching networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019, pp. 6859–6866. doi:10.1609/aaai.v33i01.33016859.

[15] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: A method for automatic evaluation of machine translation, in: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, Association for Computational Linguistics, USA, 2002, p. 311–318. URL: https://doi.org/10.3115/1073083.1073135. doi:10.3115/1073083.1073135.

[16] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, Y. Artzi, Bertscore: Evaluating text generation with bert, 2020. arXiv:1904.09675.

[17] C. Rebuffel, T. Scialom, L. Soulier, B. Piwowarski, S. Lamprier, J. Staiano, G. Scoutheeten, P. Gal-

linari, Data-QuestEval: A Reference-less Metric for Data-to-Text Semantic Evaluation, in: 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Punta Cana, Dominican Republic, 2021, pp. 8029–8036. URL: https://hal.sorbonne-universite.fr/hal-03479905. doi:10.18653/v1/2021.emnlp-main.633.

[18] K. Zhang, O. Balalau, I. Manolescu, 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, 2023. URL: https://hal.science/hal-04257838.

[19] P. Ke, H. Ji, Y. Ran, X. Cui, L. Wang, L. Song, X. Zhu, M. Huang, 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, 2021, pp. 2526–2538. doi:10.18653/v1/2021.findings-acl.223.

[20] L. F. R. Ribeiro, M. Schmitt, H. Schütze, I. Gurevych, Investigating pretrained language models for graph-to-text generation, in: Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI, Association for Computational Linguistics, Online, 2021, pp. 211–227. URL: https://aclanthology.org/2021.nlp4convai-1.20. doi:10.18653/v1/2021.nlp4convai-1.20.

[21] L. Nan, D. Radev, R. Zhang, A. Rau, A. Sivaprasad, C. Hsieh, X. Tang, A. Vyas, N. Verma, P. Krishna, Y. Liu, N. Irwanto, J. Pan, F. Rahman, A. Zaidi, M. Mutuma, Y. Tarabar, A. Gupta, T. Yu, Y. C. Tan, X. V. Lin, C. Xiong, R. Socher, N. F. Rajani, 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, Association for Computational Linguistics, 2021, pp. 432–447. doi:10.18653/v1/2021.naacl-main.37.

[22] T. Castro Ferreira, C. Gardent, N. Ilinykh, C. van der Lee, S. Mille, D. Moussallem, A. Shimorina, 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+), Association for Computational Linguistics, Dublin, Ireland (Virtual), 2020, pp. 55–76. URL: https://aclanthology.org/2020.webnlg-1.7.

[23] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, Journal of Machine Learning Research 21 (2020) 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[24] N. Reimers, I. Gurevych, 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, 2019. URL: https://arxiv.org/abs/1908.10084.

[25] H. Shimada, H. Nakagawa, T. Tsuchiya, Goal model construction based on user review classification, in: REFSQ Workshops, 2019. URL: https://api.semanticscholar.org/CorpusID:186206199.

[26] H. Nakagawa, H. Shimada, T. Tsuchiya, Interactive goal model construction based on a flow of questions, IEICE Trans. Inf. Syst. 103-D (2018) 1309–1318. URL: https://api.semanticscholar.org/CorpusID:145941947.

[27] K. Watanabe, H. Nakagawa, T. Tsuchiya, Kaos modeling editor: A tool for semi-automated goal modeling (2023).

[28] M. Shibaoka, H. Kaiya, M. Saeki, Goore : Goal-oriented and ontology driven requirements elicitation method, in: Advances in Conceptual Modeling – Foundations and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 225–234. doi:10.1007/978-3-540-76292-8_28.