Reading Between the Lines: Information Extraction from Industry Requirements

Ole Magnus Holter
University of Oslo / Norway
olemholt@ifi.uio.no

Basil Ell
University of Oslo / Norway
Bielefeld University / Germany
basile@ifi.uio.no

Abstract

Industry requirements describe the qualities that a project or a service must provide. Most requirements are, however, only available in natural language format and are embedded in textual documents. To be machine-understandable, a requirement needs to be represented in a logical format. We consider that a requirement consists of a scope, which is the requirement's subject matter, a condition, which is any condition that must be fulfilled for the requirement to be relevant, and a demand, which is what is required. We introduce a novel task, the identification of the semantic components scope, condition, and demand in a requirement sentence, and establish baselines using sequence labelling and few-shot learning. One major challenge with this task is the implicit nature of the scope, often not stated in the sentence. By including document context information, we improved the average performance for scope detection. Our study provides insights into the difficulty of machine understanding of industry requirements and suggests strategies for addressing this challenge.

1 Introduction

Requirements are a critical part of the development process for products and services. They are documented descriptions of the physical or functional qualities that a product or a service must have. In industry, requirements serve as a means of communication between contractors and manufacturers, defining what is expected to be built and the quality standards to be met. Governments and international organizations may also impose requirements to ensure compliance with rules, regulations and standards. Requirements are included as part of the contract between two parties, making adherence to them a legal obligation. Violating the requirements can lead to legal implications and financial losses, underscoring the importance of careful specification and adherence to requirements throughout the development process.

A requirement is typically associated with a specific piece of equipment that needs to be built which is referred to as the scope of the requirement. However, a requirement may only be relevant if certain conditions are met, which will be referred to as condition. The demand of the requirement is a feature or quality the scope must possess. As an example, consider the requirement *equipment with a weight of more than 1000 kg shall have a weight certificate*. Here, the scope is *equipment*, while the condition is *with a weight of more than 1000 kg*, and the demand is to *have a weight certificate*.

Most requirements are expressed as natural language text and are embedded in documents. These documents often have a hierarchical structure with chapters, sections, and other headings, which provide important context for understanding the requirements. When the number of requirements documented in this way increases, managing and maintaining these documents becomes a significant challenge. In addition, checking for consistency in a set of requirements and ensuring compliance of project descriptions with a set of requirements are time-consuming tasks that ideally should be automated. To overcome these challenges, computer systems could be used to automatically identify relevant requirements, check consistency and ensure compliance. This could be achieved by creating all new requirements in a machine-understandable format. However, the industry is often bound by existing requirements in their current form. Therefore, extracting information from existing documents is essential for enabling automated systems.

We have found that identifying the scope of a requirement can be particularly challenging since this information is often not explicitly stated in the sentence. Context and additional information is often required to make accurate predictions. The document's title, section headings and domain knowledge can provide valuable context for identifying the scope of a requirement.

Proceedings of Recent Advances in Natural Language Processing, pages 703–711
Varna, Sep 4–6, 2023
https://doi.org/10.26615/978-954-452-092-2_076

Recent studies have shown that adding contextual information can improve the performance of NLP tasks that typically focus on one sentence at a time. For instance, in named entity recognition (NER), Wang et al. (2021) used unstructured text as context, while Shahzad et al. (2021) incorporated image-based information. Similarly, context has been found to be beneficial in relation extraction. E.g., Bastos et al. (2021) improved performance using information from knowledge graphs.

Our work makes three key contributions. First, it introduces a novel task: identifying three semantic components `scope`, `condition`, and `demand` of a requirement sentence. Second, it establishes baselines for this task. Third, it investigates the extent to which including context information from the document can improve the quality of identifying these semantic components.

2 Related Work

Related work can be grouped into information extraction from requirements or legal text, and using context information to improve sequence labelling.

2.1 Information Extraction from Requirements

A substantial amount of work has been done with natural language processing (NLP) techniques on requirements. Much of this is in the area of software requirements where most studies have focused on analysing and improving requirements. For an overview of approaches and techniques used on software requirements, see (Zhao et al., 2022). Relatively little work, however, has been done on information extraction from software requirements. One work, by Schlutter and Vogelsang (2020), uses semantic role labelling to model software requirements as RDF graphs for semantic search. The CiRA tool classifies a requirement into causal and not causal and identifies causal clauses (Fischbach et al., 2021). One of the major challenges in dealing with requirements is that they are typically copyrighted and cannot be shared. Thus, comparing the performance of tools is a challenge. The PURE dataset, a collection of software requirement documents, was proposed by Ferrari et al. (2017). The dataset has been labelled for and used to distinguish requirement sentences from other types of text in requirement documents using a BERT-based classifier (Ivanov et al., 2022). Another dataset was proposed by Fischbach et al. (2020).

Some work has also been done on industry requirements. Fantoni et al. (2021) suggested that syntactic and morphological rules together with ontologies can be used to classify parts of a project description into subprojects in the railway industry. An NLP pipeline was used to extract concepts from the technical requirements about IBM Thinkpad Laptops and to link concepts to a knowledge base (Vierlboeck et al., 2022). A similar approach was proposed, but using lexical and syntactical rules for the extraction of semantic roles of 300 sentences, in (Fritz et al., 2021). Weak supervision and a BERT-based model were used to identify which requirement sentences mention the requirements' subject matter (`scope`) and which do not mention it (Holter and Ell, 2021).

2.2 Information Extraction from Legal Text

Legal text has many similarities with requirements. The language is domain-specific and the documents may have some structure (i.e., headers, subheaders). In addition, some tasks that one wants to solve on legal text are often similar to what we would like to solve for requirements. On legal text, it has been demonstrated that pretraining the BERT model on a corpus of domain-specific texts can improve the performance on several downstream tasks (Limsopatham, 2021; Elwany et al., 2019). They also demonstrate that RoBERTa performs better than BERT and that the performance is relatively good on the tasks even if it is trained on a general corpus only. A combination of deep semantic parsing and manual rules was used to identify normative clauses (obligations, permissions, prohibitions) from legal text by Dragoni et al. (2016). Ferraro et al. (2019) identify challenges when working with legal text and outlines a possible strategy for the automatic extraction of normative rules. In (Michel et al., 2022), the authors use FastText and a convolutional network to identify decision rules.

2.3 Using Context Information

It has been demonstrated that context information helps to improve the performance of some NLP systems. Often, a knowledge base is used, but context information can come from various sources. Liu et al. (2020) show that the BERT model improves performance on multiple tasks when including information from knowledge bases. For relation extraction, using context information from knowledge bases was found to improve performance (Bastos et al., 2021; Nadgeri et al., 2021). Wang et al.

(2021) noted that unstructured text retrieved from a search engine improves performance on NER. Incorporating images was shown to improve performance when doing named entity recognition on social media posts in (Shahzad et al., 2021).

3 Preliminaries

3.1 Semantic Modelling of Requirements

In previous work by Klüwer and DNV GL (2019) and in ISO 15926-14 (Walther et al., 2020), a requirement $\mathcal{R}$ is defined as a logical axiom which stipulates that if x belongs to a class $\mathcal{S}$ and satisfies a condition $\mathcal{C}$ (may be empty), then $\mathcal{R}$ is satisfied only when the demand $\mathcal{D}$ is also true. This relationship can be expressed in first-order logic as:

$$\forall x((\mathcal{S}(x) \wedge \mathcal{C}(x)) \rightarrow \mathcal{D}(x))$$

To formalize a requirement $\mathcal{R}$ expressed in natural language, such as *Equipment made of metal exposed to seawater shall have an anti-corrosive coating*, we can express that for any object x belonging to the class `Equipment` and satisfying a condition `exposedToSeawater`, and if x is made of the material y belonging to the class `Metal`, there shall exist a feature u such that x has that feature and u belongs to the class `AntiCorrosiveCoating`.

```
∀x∀y∀z∃u(Equipment(x)
    ∧ madeOf(x,y) ∧ Metal(y)
    ∧ exposedToSeawater(x)
  → hasFeature(x,u) ∧ AntiCorrosiveCoating(u))
```

3.2 Problem Description

In the context of this work, a sentence is a sequence of words where the first word starts with a capital letter and the sequence ends with a period. A requirement sentence is a type of sentence that expresses a demand or a feature that a piece of equipment must have to conform to the specifications outlined in the document, and possibly a condition. Let R be a set of requirement sentences and r be a requirement sentence. We define three sets S, C, and D to represent the textual representations of `scope`, `condition`, and `demand`, respectively.

The task that we introduce in this paper is to realize a function $f : R \rightarrow \mathcal{P}(S) \times \mathcal{P}(C) \times \mathcal{P}(D)$ that predicts a triple on the form (S', C', D') where $S' \subseteq S$, $C' \subseteq C$, $D' \subseteq D$. Thus, given a requirement r, the function returns a set S' of scopes, a set C' of conditions, and a set D' of demands, i.e., $f(r) = (S', C', D')$.

3.3 Identification of the `scope`

The `scope` refers to the requirement's subject matter, such as specific components or systems. Identifying the `scope` of a requirement can be challenging, as it may not be explicitly stated, but implied from the document context. For example, in the requirement RU-HSLC-Pt5-Ch6 Section 3 SAFETY REQUIREMENT [3.9.5] (Sent. 1) *The system need not be designed with redundancy in pumps or back-up pressure tank*, the context reveals that it is about *Accommodation sprinkler system*, even though the sentence uses a general term (i.e., system).

3.4 Identification of the `condition`

The `condition` refers to a condition that must be fulfilled for the requirement on the `scope` to be relevant. It may be a direct property of the equipment, as in *Equipment with weight more than 500 kg*, or it may be related to some process associated with the `scope`. As opposed to the `scope`, the `condition` is typically explicit in the sentence.

3.5 Identification of the `demand`

The `demand` is the essential requirement expressed in the sentence. It defines what is needed for the `scope` (under the specified `condition`) to conform to the specifications outlined in the agreement. Typically, a requirement sentence will contain the demand explicitly, and it often constitutes a substantial part of the sentence. For instance, consider the requirement *Equipment made of metal exposed to seawater shall have an anti-corrosive coating*. The demand would be *have an anti-corrosive coating*.

4 Method

4.1 Dataset Creation

We utilized 23 PDF documents from Det Norske Veritas (DNV),[1] an international company specializing in classification and risk management. All documents are written in English and were obtained from DNV's website.[2] We extracted the text using Apache PDF box (v2.0.1) and used regular expressions to identify the document structure such as headers, sub-headers, and figures. We then created a semi-structured XML version of the PDFs. Sentence tokenization was achieved using spaCy.[3]

[1] All documents are copyrighted ©DNV. DNV does not take responsibility for any consequences arising from the use of this content.

[2] From https://rules.dnv.com/ 2022.9.21

[3] spaCy v3.4.1 with en_core_web_sm v3.4.0

To identify requirement sentences, we extracted only those sentences containing the word "shall." According to DNV documents, "shall" is a verbal form denoting "requirements strictly to be followed" (Det Norske Veritas, Ed. July 2022). While there are some sentences containing "shall" that are not requirements (e.g., definitions), we did not observe any requirements without "shall." From each of the semi-structured XML documents, we randomly extracted 100 such requirement sentences, resulting in a set of 2225 requirement sentences.

We use two types of context information. First, we followed the document tree of the XML file from the document title down to a single sentence, concatenating the headers to the sentence, separated with a dot. Second, we concatenated the noun chunks of all sentences with the same requirement number, after the headers, also separated by a dot.

Finally, we manually annotated the resulting strings using prodigy[4] for the spans of the `scope`, the `condition` and the `demand`. For an overview of the data-creation process, see Figure 1.

We developed an annotation guideline to ensure a consistent annotation process. The first author followed the guideline and annotated the data to create the gold standard. That author consulted original documents to see a requirement within its original context whenever necessary. A subset of the annotations was validated by the second author.

During the annotation process, we discarded 78 (about 3 %) of the extracted sentences because they were improperly extracted from the documents, resulting in incomplete or fragmentary sentences.

We utilized a token-level annotation scheme where a token may be assigned one of three possible labels: `scope`, `condition`, `demand`. If a `scope` occurred within a `condition` or a `demand`, we labelled it as `scope` instead of both as `scope` and `condition` or `demand`, to maintain consistency and enable the use of a labelling scheme that does not support multiple labels per token.

4.2 Dataset Overview

The final dataset contained a total of 2147 requirement sentences. We created two datasets from the original dataset: one with contextual information, including titles, header information, and surrounding noun chunks, as described above (Train^c), and one without this context (Train). Table 1 show the number of spans for each label in the two datasets and the number of sentences containing at least one label of each type. Notably, only about half of the sentences in Train have a `scope` label.

Label	Train^c	STrain^c	Train	STrain
`scope`	4862	2074	1333	1017
`condition`	733	620	713	609
`demand`	3895	2147	3836	2135

Table 1: Distribution of labels for the Train^c and Train. STrain^c and STrain count how many sentences have at least one label of the type.

4.3 Sequence Labelling

As a sequence labelling model, we used RoBERTa.[5] To train and evaluate the model, and estimate the effect of data split, we performed 5-fold cross-validation on both Train^c and the Train datasets. Thus, we trained five models for each dataset with slightly different data. We utilized a RoBERTa model (Liu et al., 2019) with a classification layer. We fine-tuned the `roberta_for_token_classification` model from the HuggingFace library (Wolf et al., 2019). The hyperparameters[6] were adapted from fine-tuning experiments in the RoBERTa paper (Liu et al., 2019) and we did no parameter tuning.

To obtain a textual representation of the spans labelled with `scope`, we retrieved the sequences of contiguous tokens with this label generated by the model. We then post-processed these spans to remove duplicates such as *Equipment* and *equipment*. Post-processing included removing the definite article, case normalization of all tokens in the chunk, removal of extra spaces and punctuations and removing unmatched parentheses. We used spaCy[7] for tokenization and regular expressions for the post-processing. Note that a single sentence can contain multiple `scope` spans, which we collect in a set to merge exact duplicates.

Similarly, we extracted the `condition` and `demand` spans, but did not remove the definite articles because for `condition` and `demand`, we expected to extract sub-sentences and not concepts.

4.4 Few-shot

Alternatively, the problem can be approached as a language generation task. In this case, we adopt a

[4]Prodigy v1.11.8

[5]RoBERTa large 355M parameters

[6]lr=1e-5, optimizer=adamW, epochs=4, dropout=0.5

[7]spaCy v3.4.3 with en_core_web_sm v3.4.1

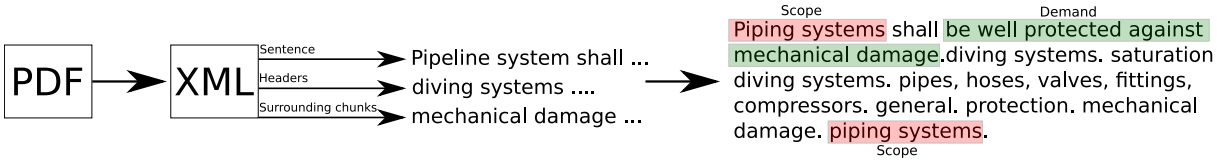

Figure 1: Overview of the data creation and labelling process. Example from OS-E402 Chapter 3 SATURATION DIVING SYSTEM Section 7 1.5.

few-shot learning approach by designing a simple prompt. We incorporate 10 examples from either the Train or Train^c annotated dataset and create input-output pairs. To ensure relevant examples, we employ the `all-roberta-large-v1` from the `sentence-transformers` library (Reimers and Gurevych, 2019) to select the 10 most semantically similar instances for each target sentence and incorporate these into the prompt.

In this approach, the desired output is a JSON document that includes the elements: `scope`, `condition`, and `demand`. We prompt GPT-3[8] and retrieve the `scope`, `condition`, and `demand` as provided by the model to obtain the textual representation of the semantic components.

By comparing the sequence labelling and the few-shot learning approach, we obtain a better understanding of their respective strengths and limitations, providing insights into which approach might be more suitable for this particular task.

5 Evaluation

To assess the generalization capabilities of our approaches across different domains and evaluate their performance on unseen documents, we extracted and labelled an additional 400 sentences extracted from four other documents. Two documents were selected from the same domain (*High speed and light craft*), while the other two were from different domains: *Floating fish farming units and installation* and *Drilling facilities*.

Consistent with our previous datasets, we labelled the sentences and created two versions of each dataset: one with context (OS-E101^c, RU-HSLC-Pt5^c, RU-HSLC-Pt6^c, OU-0503^c), and one without context (OS-E101, RU-HSLC-Pt5, RU-HSLC-Pt6, OU-0503). This differentiation is necessary because the model trained with context expects an input format where the sentence has headers and noun chunks appended, while the model trained without context expects the sentence only.

[8]OpenAI's text-davinci-003 175 billion parameters

To evaluate the performance of the models on `scope`, `condition`, and `demand` detection, we annotated each sentence in the corpus with the textual representation of `scope`, `condition`, and `demand`. We used the original documents as a guide to ensure accurate annotations. The gold `scope` comprises a set of normalized noun chunks, which we combined into a set as described in Section 4. Note that the gold `scope`, `condition`, and `demand` are the same both for the dataset with context and the dataset without context.

We evaluate the performance using three different metrics used in the text generation literature: ROUGE-L (Lin, 2004), BLEU Unigrams (Papineni et al., 2002), and a language-model-based measure (LBM). To use similarity measures from text generation literature, we created a single string from the sets of extracted `scope`, `condition`, and `demand` spans by joining items with the word "and." Some of the metrics do not handle empty strings, therefore, if the predicted string or the gold string is empty, we replace it with a dummy string "EMPTY." We then compare the predicted and gold strings using the respective similarity scores.

For the LBM metric, we utilized sentence embeddings generated by the `all-roberta-large-v1` model from the `sentence-transformers` library (Reimers and Gurevych, 2019). The embedding captures the semantic meaning of sentences and enables us to estimate the semantic similarity between predicted and gold strings. The LBM score is the cosine similarity between the predicted and gold strings.

5.1 Establishing a Lower Bound

To establish a lower bound for comparison, we evaluated a "model" that is expected to perform poorly on the task. While this approach is deliberately simplistic, it is not maximally naive and incorporates some level of reasonableness. The predictions of this baseline model are as follows: i) The predicted `scope` is the generic term *component*, which is a term that appears among the scopes in

the dataset. The choice is motivated by the fact that many equipment items mentioned in the requirements are components of a larger system. ii) The predicted `condition` is an empty set, as the majority of the requirements have no conditions. iii) The predicted `demand` is the entire sentence itself, as the demand is often a significant portion of the sentence. We applied the evaluation metrics of the output of this model and report the results in Table 2. This provides a point of comparison for assessing the performance of other models on the task using the same metrics.

Note that the purpose of this baseline approach is to establish a reference point for evaluating the relative performance of more advanced models.

Document	ROUGE	BLEU-1	LBM
`scope`	0.03	0.01	0.22
`condition`	0.69	0.69	0.72
`demand`	0.47	0.25	0.66

Table 2: Evaluation of the performance of the model that uses *component* as `scope`, an empty `condition` and the whole sentence as `demand` on the test split.

5.2 Without Context

We conducted 5-fold cross-validation on the training data without context (Train) following the methodology outlined in Section 4. Then, we measure the performance of each model on each fold's test data and the four other documents. We then compared the extracted spans with the gold spans and report the results in Table 3.

Regarding `scope` detection as sequence labelling task, the RoBERTa model achieved an average ROUGE-L F1 score of 0.45. However, in the few-shot learning approach using GPT-3, we observed superior performance with an average ROUGE-L F1 score of 0.57. For `condition` detection, the RoBERTa model achieved the highest average ROUGE-L F1 score of 0.88; outperforming GPT-3. In terms of `demand` detection, the RoBERTa yielded an average ROUGE-L F1 score of 0.78, outperformed by the GPT-3 with a score of 0.85. However, the sequence labelling approach obtained a higher LBM score than GPT-3.

5.3 With Context

Similarly, we conducted 5-fold cross-validation on the training data with context (Train^c). Subsequently, we utilized the five models to predict the spans of `scope`, `condition`, and `demand` on the Test^c data and the four additional documents. The predicted spans were compared against the gold spans, and the results are presented in Table 4. In terms of `scope` and `condition` extraction, the RoBERTa sequence labeller outperformed the few-shot approach. However, when it comes to `demand` extraction, the RoBERTa model and the few-shot approach demonstrated similar performance.

6 Discussion

The sequence labelling model achieves a ROUGE-L F1 score of 0.45 on `scope` detection without access to context information. Considering that only half of the sentences in the Train dataset have `scope` labels, the performance is promising. GPT-3, being much larger, is able to leverage information learned during pre-training and generate scopes that are not explicitly mentioned in the text, allowing improved performance compared to the sequence labelling model.

The sequence labelling approach demonstrates strong performance in the detection of the `condition` and `demand`. The results suggest that with a larger training corpus, the accuracy is likely to be suitable for practical applications.

The challenge with `scope` detection lies in the need to infer implicit scopes by "reading between the lines." To address this challenge, our study proposes the explicit inclusion of context information to enhance performance. By incorporating document context, for most sentences, we have `scope` labels. Either, the labels come from the sentence itself or from the context, as seen in Table 1.

On `scope` detection with context, we observe a general improvement over the results without context. Despite the increased number of sentences with `scope` labels in the training data, the improvement achieved by the models does not align proportionally. In particular, GPT-3 does not effectively leverage context information. It is possible that presenting the examples differently or refining the prompt could lead to improved results.

The observed improvement with context information is most prominent on the test data and on OU-0503. Performance improvements on test data may in part be explained by the model learning relevant terms used in headers in the documents used for training. However, the improvement on OU-0503 demonstrates that the model is still able to generalize, and not only memorise `scope` labels.

Document	scope			condition			demand		
	ROUGE	BLEU-1	LBM	ROUGE	BLEU-1	LBM	ROUGE	BLEU-1	LBM
RoBERTa large									
Test	0.38±0.02	0.36±0.02	0.48±0.02	0.88±0.03	0.87±0.04	0.89±0.02	0.78±0.04	0.80±0.07	0.90±0.01
OS-E101	0.52±0.05	0.50±0.06	0.61±0.05	0.89±0.02	0.87±0.03	0.91±0.02	0.78±0.03	0.79±0.08	0.91±0.01
RU-HSLC-Pt5	0.50±0.03	0.46±0.06	0.59±0.04	0.87±0.02	0.86±0.03	0.89±0.02	0.82±0.07	0.82±0.09	0.92±0.02
RU-HSLC-Pt6	0.45±0.05	0.45±0.05	0.55±0.04	0.90±0.04	0.88±0.05	0.91±0.03	0.78±0.07	0.79±0.11	0.90±0.03
OU-0503	0.40±0.05	0.38±0.07	0.50±0.05	0.87±0.03	0.86±0.04	0.88±0.03	0.74±0.07	0.78±0.09	0.91±0.02
Average	0.45	0.43	0.55	**0.88**	**0.87**	**0.90**	0.78	0.80	**0.91**
GPT-3 10-shot									
Test (100 unseen)	0.66	0.65	0.74	0.83	0.81	0.84	0.84	0.85	0.90
OS-E101	0.51	0.47	0.64	0.75	0.73	0.77	0.85	0.83	0.89
RU-HSLC-Pt5	0.63	0.57	0.70	0.81	0.80	0.83	0.89	0.88	0.93
RU-HSLC-Pt6	0.58	0.57	0.71	0.76	0.75	0.79	0.82	0.81	0.88
OU-0503	0.47	0.42	0.60	0.79	0.77	0.79	0.85	0.83	0.89
Average	**0.57**	**0.54**	**0.68**	0.79	0.77	0.81	**0.85**	**0.84**	0.90

Table 3: Results of detecting scope, condition, and demand without context. Measured using ROUGE-L F1, BLEU-1, and LBM cosine similarity. Values are averages, with confidence intervals (from 5-fold experiments).

Document	scope			condition			demand		
	ROUGE	BLEU-1	LBM	ROUGE	BLEU-1	LBM	ROUGE	BLEU-1	LBM
RoBERTa large									
Test^c	0.72±0.04	0.68±0.04	0.82±0.04	0.88±0.02	0.87±0.02	0.89±0.01	0.81±0.02	0.82±0.10	0.90±0.02
OS-E101^c	0.67±0.04	0.60±0.03	0.74±0.02	0.90±0.01	0.89±0.01	0.92±0.01	0.81±0.05	0.82±0.09	0.91±0.04
RU-HSLC-Pt5^c	0.67±0.00	0.59±0.02	0.77±0.01	0.87±0.04	0.86±0.05	0.89±0.03	0.86±0.03	0.86±0.11	0.92±0.03
RU-HSLC-Pt6^c	0.69±0.05	0.65±0.05	0.76±0.04	0.91±0.02	0.90±0.02	0.92±0.02	0.82±0.04	0.84±0.09	0.90±0.03
OU-0503^c	0.72±0.05	0.56±0.05	0.76±0.04	0.88±0.04	0.88±0.04	0.90±0.03	0.77±0.06	0.79±0.10	0.89±0.03
Average	**0.70**	**0.62**	**0.77**	**0.89**	**0.88**	**0.90**	0.81	**0.83**	**0.90**
GPT-3 10-shot									
Test^c (100 unseen)	0.71	0.71	0.79	0.80	0.79	0.82	0.84	0.85	0.90
OS-E101^c	0.60	0.56	0.70	0.73	0.72	0.76	0.82	0.79	0.88
RU-HSLC-Pt5^c	0.63	0.58	0.70	0.75	0.73	0.77	0.89	0.83	0.92
RU-HSLC-Pt6^c	0.66	0.64	0.75	0.80	0.79	0.82	0.84	0.83	0.89
OU-0503^c	0.72	0.57	0.77	0.78	0.77	0.81	0.86	0.84	0.90
Average	0.66	0.61	0.74	0.77	0.76	0.80	**0.85**	**0.83**	**0.90**

Table 4: Results of detecting scope, condition, and demand with context. Measured using ROUGE-L F1, BLEU-1, and LBM cosine similarity. Values are averages, with confidence intervals (from 5-fold experiments).

Including context from the document did not result in improvements for condition and demand detection. The difference between the experiments with and without context is consistently small.

7 Conclusion and Future Work

We have introduced the novel task of identifying the semantic components scope, condition, and demand in a requirement sentence. We have established baselines by casting the task as a sequence labelling problem and a few-shot learning problem. We have also highlighted the particular challenge of identifying the scope which is often not explicitly given and proposed including context information explicitly to improve scope detection.

Including context information in the text is helpful for identifying the scope of a requirement sentence in all the requirements documents in this experiment. In addition, this work establishes that the detection of scope is very different from the detection of a condition and the demand, and that different approaches work differently for scope detection than for condition and demand detection. It may thus be useful to consider them as different tasks, requiring different tools and strategies.

In future work one could $i)$ investigate when adding context helps, $ii)$ investigate what kind of context helps, or $iii)$ investigate other types of context information and how to present the context information to a language model. Furthermore, $iv)$ matching the scopes to concepts in a knowledge graph would be interesting as thereby it could be possible to resolve similar textual representations of the same ontological concept. Finally, $v)$ more research is needed to see if our results also apply to requirements from other sources.

Ethics Statement

While we do not think this study poses any risks, a system that performs automatic compliance checking of requirements must be sound and complete, to not reject designs that satisfy all relevant requirements or not accept a design that does not satisfy all relevant requirements. Falsely rejecting a valid design can lead to financial losses for the company whose design was rejected, and falsely accepting an invalid design can cause dangers. More effective requirement management would give a company a competitive advantage. However, this can lead to other skills being required of employees.

Acknowledgements

This study is funded by the SIRIUS centre[9]: Norwegian Research Council project number 237898. It is co-funded by partner companies, including DNV and Equinor. We would also like to thank Christoph Leiter for helping with the evaluation.

References

Anson Bastos, Abhishek Nadgeri, Kuldeep Singh, Isaiah Onando Mulang, Saeedeh Shekarpour, Johannes Hoffart, and Manohar Kaul. 2021. Recon: relation extraction using knowledge graph context in a graph neural network. In *Proceedings of the Web Conference 2021*, pages 1673–1685.

Det Norske Veritas. Ed. July 2022. RULES FOR CLASSIFICATION: Ships. Technical report, DNV-RU-SHIP. ©DNV GL.

Mauro Dragoni, Serena Villata, Williams Rizzi, and Guido Governatori. 2016. Combining NLP approaches for rule extraction from legal documents. In *1st Workshop on MIning and REasoning with Legal texts (MIREL 2016)*.

Emad Elwany, Dave Moore, and Gaurav Oberoi. 2019. BERT goes to law school: Quantifying the competitive advantage of access to large legal corpora in contract understanding. In *Workshop on Document Intelligence at NeurIPS 2019*.

Gualtiero Fantoni, Elena Coli, Filippo Chiarello, Riccardo Apreda, Felice Dell'Orletta, and Guido Pratelli. 2021. Text mining tool for translating terms of contract into technical specifications: Development and application in the railway sector. *Computers in Industry*, 124:103357.

Alessio Ferrari, Giorgio Oronzo Spagnolo, and Stefania Gnesi. 2017. PURE: A Dataset of Public Requirements Documents. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 502–505. IEEE.

Gabriela Ferraro, Ho-Pun Lam, Silvano Colombo Tosatto, Francesco Olivieri, Mohammad Badiul Islam, Nick van Beest, and Guido Governatori. 2019. Automatic extraction of legal norms: Evaluation of natural language processing tools. In *JSAI International Symposium on Artificial Intelligence*, pages 64–81. Springer.

Jannik Fischbach, Julian Frattini, Arjen Spaans, Maximilian Kummeth, Andreas Vogelsang, Daniel Mendez, and Michael Unterkalmsteiner. 2021. Automatic detection of causality in requirement artifacts: the cira approach. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 19–36. Springer.

Jannik Fischbach, Benedikt Hauptmann, Lukas Konwitschny, Dominik Spies, and Andreas Vogelsang. 2020. Towards causality extraction from requirements. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 388–393. IEEE.

Simon Fritz, Vethiga Srikanthan, Ryan Arbai, Chenwei Sun, Jivka Ovtcharova, and Hendro Wicaksono. 2021. Automatic information extraction from text-based requirements. *International Journal of Knowledge Engineering*, 7(1).

Ole Magnus Holter and Basil Ell. 2021. Towards Scope Detection in Textual Requirements. In *3rd Conference on Language, Data and Knowledge (LDK 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Vladimir Ivanov, Andrey Sadovykh, Alexandr Naumchev, Alessandra Bagnato, and Kirill Yakovlev. 2022. Extracting Software Requirements from Unstructured Documents. *arXiv preprint arXiv:2202.02135*.

Johan W Klüwer and DNV GL. 2019. OWL Upper Ontology for Reified Requirements. https://data.dnv.com/ontology/requirement-ontology/documentation/req-ont.pdf accessed: 2023-01-13.

Nut Limsopatham. 2021. Effectively leveraging BERT for legal document classification. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 210–216. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: Enabling Language Representation with Knowledge Graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2901–2908.

[9] http://sirius-labs.no

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Maximilian Michel, Djordje Djurica, and Jan Mendling. 2022. Identification of decision rules from legislative documents using machine learning and natural language processing. In *HICSS*, pages 1–10.

Abhishek Nadgeri, Anson Bastos, Kuldeep Singh, Isaiah Onando Mulang', Johannes Hoffart, Saeedeh Shekarpour, and Vijay Saraswat. 2021. KGPool: Dynamic knowledge graph context selection for relation extraction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 535–548. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics.

Aaron Schlutter and Andreas Vogelsang. 2020. Knowledge Extraction from Natural Language Requirements into a Semantic Relation Graph. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 373–379.

Moemmur Shahzad, Ayesha Amin, Diego Esteves, and Axel-Cyrille Ngonga Ngomo. 2021. InferNER: An attentive model leveraging the sentence-level information for Named Entity Recognition in Microblogs. In *The International FLAIRS Conference Proceedings*, volume 34.

Maximilian Vierlboeck, Daniel Dunbar, and Roshanak Nilchiani. 2022. Natural Language Processing to Extract Contextual Structure from Requirements. In *2022 IEEE International Systems Conference (SysCon)*, pages 1–8.

Dirk Walther, Johan Wilhelm Klüwer, Francisco Martin-Recuerda, Arild Waaler, Daniel Lupp, Maja Milicic Brandt, Stephan Grimm, Aneta Koleva, Mesbah Kahn, Lillian Hella, and Nils Sandsmark. 2020. ISO 15926 working draft proposal. `https://readi-jip.org/wp-content/uploads/2020/10/ISO_15926-14_2020-09-READI-Deliverable.pdf` accessed: 2023-01-13.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Improving named entity recognition by external context retrieving and cooperative learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1800–1812, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Liping Zhao, Waad Alhoshan, Alessio Ferrari, and Keletso J. Letsholo. 2022. Classification of Natural Language Processing Techniques for Requirements Engineering. *arXiv preprint arXiv:2204.04282*.