Towards Scope Detection in Textual Requirements

Ole Magnus Holter
Department of Informatics, University of Oslo, Norway

Basil Ell
Department of Informatics, University of Oslo, Norway

Abstract

Requirements are an integral part of industry operation and projects. Not only do requirements dictate industrial operations, but they are used in legally binding contracts between supplier and purchaser. Some companies even have requirements as their core business. Most requirements are found in textual documents, this brings a couple of challenges such as ambiguity, scalability, maintenance, and finding relevant and related requirements. Having the requirements in a machine-readable format would be a solution to these challenges, however, existing requirements need to be transformed into machine-readable requirements using NLP technology. Using state-of-the-art NLP methods based on end-to-end neural modelling on such documents is not trivial because the language is technical and domain-specific and training data is not available. In this paper, we focus on one step in that direction, namely scope detection of textual requirements using weak supervision and a simple classifier based on BERT general domain word embeddings and show that using openly available data, it is possible to get promising results on domain-specific requirements documents.

2012 ACM Subject Classification Computing methodologies → Natural language processing

Keywords and phrases Scope Detection, Textual requirements, NLP

Digital Object Identifier 10.4230/OASIcs.LDK.2021.31

Supplementary Material *Software (Source Code)*: `https://github.com/oholter/LDK2021_toward_scope_detection`; archived at `swh:1:dir:ea53d4ebc3fb613c158bcf53bc31a86e55f30e99`

Funding This research is funded by the SIRIUS centre: Norwegian Research Council project number 237898. It is co-funded by partner companies including DNV GL and Equinor.

1 Introduction

The number of standards, requirements, and recommended practices (RP) in industry is overwhelming and requirements administration is costly. A survey from 2016 reveals that costs related to requirements are particularly high for petroleum companies working on the Norwegian shelf [23]. There are international standards like ISO, IEC, national standards, industry standards and companies have their own set of standards and requirements and all of them are constantly evolving.

Most of the requirements that are used in industry today are available only in textual format embedded in documents (e.g., PDF, Word) and the documents are revised and treated as one entity. This way of dealing with requirements comes with numerous challenges. One challenge is the organization of the requirements (i.e., managing the documents). When a supplier agrees to build a specific artefact, he must know exactly what requirements are relevant for the task and to find them, he must look into many potentially relevant documents. The documents may contain duplicate, overlapping, and even conflicting requirements. The revision cycle time of requirements is also a major bottleneck for industry today. Before an international standard has been revised, the world has changed and they already seem outdated. This slow revision cycle is one of the main drivers of company-specific requirements which in turn creates a more complicated picture by multiplying the number of possible requirements, duplicates, and conflicts.

Some companies have migrated to requirement management systems such as SIEMENS' Polarion [30] and IBM's DOORS [21]. These tools allow for better organization of textual requirements. By adopting a proper numbering scheme where a requirement can be identified globally across documents and allowing metadata to be attached to a single requirement (e.g., author, comments), the requirements can be revised individually. This is a step in the right direction, but we are still dealing with requirements in natural language and not with machine-understandable requirements.

It would be of great help for industrial companies and requirements companies if the requirements were available in a structured, machine-understandable, format (e.g., RDF). That way, a machine will know what entity or process the requirement is about and also the relevant properties, conditions, and demands without having to figure it out by itself by interpreting the text. If we consider this requirement sentence: *Test pieces for transverse weld (cross weld) tensile shall be rectangular and in accordance with [B.2.3.3].* It can be broken down into its scope, and the demands:

- `Scope:` *Test pieces for transverse weld (cross weld)*
- `Demand 1:` *rectangular*
- `Demand 2:` *in accordance with [B.2.3.3].*

If every document is broken down into individual requirements and each requirement into its components, and if each of the components is linked with corresponding concepts in a knowledge base, requirements management can be done much more automatically because the requirements could also be used directly by computer applications. Revision cycles could be greatly reduced by not having to revise an entire document, it will be possible to detect duplicates and conflicts. We further imagine that relevant requirements for a particular project can be automatically identified, as well as which requirement belongs to which subset of the project. It can also be possible to do automatic compliance checking between a set of requirements and a project description. While new requirements can be made structural if industry decides on a standard model to represent requirements, we are still bound by over a century of textual requirements and standards that cannot be ignored.

Neural methods in NLP have improved performance on many NLP tasks. Such methods, however, require large amounts of training data. The pre-trained BERT-models, for instance, are trained using the English Wikipedia and the BookCorpus, in total about 3,300M words [16]. Working with neural NLP techniques on domain-specific documents is challenging. In the case when the available document base is small, it is not obvious how to obtain good-performing NLP-models. Different solutions have been proposed including distant supervision [25], domain specific word embeddings [26] and transfer learning approaches [29, 20]. Textual requirements documents are domain-specific, there is, to the best of our knowledge, no labelled data available, and not many publicly available knowledge bases.

Our key contribution is a method to label requirements sentences using a weakly supervised approach using openly available data and simple heuristics to train a classifier that classifies requirements sentences into **SCOPE** (contains a scope) and **NOT SCOPE** (the sentence does not contain the scope of the requirement). The source code is openly available.[1]

The remainder of this article is organized as follows. We define the problem in Section 2, before summarizing related work in Section 3. In Section sec:methodology we describe the creation of the corpus and the classifier; in Section 5 we evaluate the approach on real industry requirements before we discuss the findings in Section 6 and present our conclusions in Section 7.

[1] `https://github.com/oholter/LDK2021_toward_scope_detection`

2 The problem

Scope detection is the task of detecting a given requirement sentence's subject matter, i.e., the scope of the requirement. In this paper, we look at a binary classification task where a sentence is classified as containing a scope (**SCOPE**) or as not containing a scope (**NOT SCOPE**). According to the READI project [31, 32], a scope is typically a subclass of equipment. Thus, in this paper, we limit the task to the detection of a scope that is a piece of equipment, an assembly of pieces of equipment or refinement of equipment. A piece of equipment is understood as an artefact or a physical resource required for a purpose [5]. There can be requirement sentences with non-equipment scopes, such as about documentation, the outcome of events, quality of processes etc. However, we disregard such scopes.

We make two simplifying assumptions. First, that the scopes are in sentences with requirements that are strictly to be followed (mandatory requirements). According to [12], *shall* is a "verbal form used to indicate requirements strictly to be followed to conform to the document". For this reason, we disregard all sentences that do not contain the word *shall*. The second assumption is that the scope can be identified within a sentence, thus we disregard the surrounding context of the requirement sentence.

The requirement documents that we have been working with are structured documents using numbered sections, subsections and so on.[2] Each of the lowest level subsections can consist of either one or several sentences. Some examples of requirements sentences from the document DNVGL-ST-F101 (Submarine pipeline systems ed. October 2017) [12] are:[3]

- DNV-ST-F101 Sec.3 CONCEPT AND DESIGN PREMISE DEVELOPMENT
  3.3.4.5 *Marine growth on **pipeline systems** shall be considered, taking into account both biological and other environmental phenomena relevant for the location.*

- DNV-ST-F101 Sec.3 CONCEPT AND DESIGN PREMISE DEVELOPMENT
  3.3.6.1 *Surveys shall be carried out along the total length of the planned pipeline route to provide sufficient data for design and construction related activities.*

- DNV-ST-F101 Sec.3 CONCEPT AND DESIGN PREMISE DEVELOPMENT
  3.4.2.2 *The **submarine pipeline system** shall have a specified incidental pressure or be split into different sections with different specified incidental pressures.*

- DNV-ST-F101 Sec.4 DESIGN - LOADS
  4.2.1.7 *Fluctuations in temperature shall be taken into account when checking fatigue strength.*

The sentence in requirement **3.3.4.5** should be classified as **SCOPE** because it is about and contains a reference to *pipeline systems* which is an assembly of pieces of equipment. The sentence in **3.3.6.1** is a requirement for the planning of the pipeline route which is not a piece of equipment thus **NOT SCOPE**. **3.4.2.2** is **SCOPE** because it is about the submarine pipeline system and contains a reference to that. The sentence in **4.2.1.7** is **NOT SCOPE** since the scope is not explicit in the sentence (i.e., the fatigue strength of what?).

[2] The approach is not limited to documents with numbered sections and requirements.
[3] © DNV GL. DNV GL does not take responsibility for any consequences arising from the use of this content.

3 Related work

Scope detection of technical requirements is related to the classification of technical requirements in general. We consider four types of (sentence) classification on technical requirements: *i)* Demarcation of requirements from information, *ii)* detection of requirement defects (e.g., ambiguity, vagueness), *iii)* classifying requirements into a project's subsystem, and *iv)* classification of software requirements.

For the demarcation task, Winkler and Vogelsang propose to use a Convolutional Neural Network to classify sentences into either information or requirements using a dataset of automotive requirements that was manually labelled by industry partners [33]. Abualhaija et al. approach the same task by evaluating several statistical machine learning algorithms using a generic set of features and trained on labelled documents from different requirements domains [8]. For the detection of requirements defects, [28] uses rule-based NLP techniques (GATE). [19] tackles the project subsystem classification task by using a curated knowledge base extracted from technical specifications and a classification function, which takes into account the number of domain terms and context information. For the task of classifying software requirements using NLP techniques, a common task is the classification of software requirements into functional and non-functional requirements as in [15].

This work is also related to domain-adaption of NLP techniques such as transfer learning and weak supervision. One widely known form of weak supervision is distant supervision [24]. A typical application of distant supervision is relation extraction. To use distant supervision for relation extraction, we typically align a sentence with a database and whenever two entities are found in the sentence and there exists a relationship between the same entities in a dataset, the text is assumed to be about this relationship. Training data generated by distant supervision, however, can be noisy.

Snorkel [27] is a framework that aims to let the user programmatically build training data using a set of labelling functions. A labelling function is any code that aims to label a subset of input data. This can be simple heuristic rules or distant supervision using external knowledge bases. The user provides the framework with a set of labelling functions (LF). The framework is capable of learning a generative model to estimate the accuracy of the different labelling functions and combines the outputs into a set of probabilistic labels.

Another approach to domain adaption is transfer-learning [20]. Transfer learning means to use data or models from other tasks or domains to train a model for another task/domain [29]. An example of sequential transfer learning is to first pre-train word or sentence representations on a general corpus before fine-tuning them to the problem task/domain. Fine-tuning of pretrained BERT-embeddings for some downstream-tasks was evaluated in [16].

4 Methodology

We apply a pipeline of 5 major components as shown in Figure 1: *i)* Extraction of the textual content of the PDF *ii)* insert XML-tags to generate an XML-representation of the document, *iii)* extraction of the mandatory requirement sentences from relevant parts of the XML-document, *iv)* using data programming to create labelled training data, *v)* training of a classifier based on BERT pretrained contextual embeddings and a fully connected layer.

4.1 Notation

$\mathcal{R}$ is the set of all requirement sentences, r is a single requirement. $\mathcal{T}$ is a list of terms. **SCOPE** and **NOT SCOPE** are labels used to classify a requirement sentence as either containing the scope or not. **ABSTAIN** is used when a labelling function cannot decide.

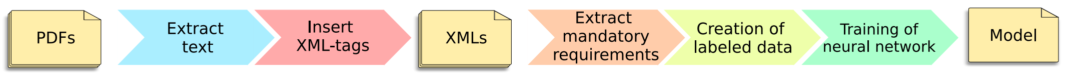

Figure 1 Overview of the pipeline.

4.2 PDF to XML

Extracting text from a PDF document is not straightforward [14]. The requirements documents we are working with are structured using numbered headings for chapters, sections, subsection, tables, figures and so on. If we extract the text from the PDF using tools such as pdftotext and work with it directly, it is often difficult to identify sentence and paragraph boundaries. We find that by first converting the text to a document where the different elements of the document are surrounded by XML-tags, we can keep both the structure and the content of the document while having a document that is easier to parse for other applications.

We use Apache PDFbox [2] to extract the text from 52 requirements documents on rules for ship classification [10] and one on submarine pipeline systems [12]. Headers and footers were removed by restricting the area that was read by PDFBox.

First, we remove dots from common abbreviations (e.g., `Sec.` was changed to `Sec`, `Ch.` to `Ch`) to make it easier to identify sentence boundaries later. Then, symbols that are part of the XML-syntax (<,> and &) were changed (to &lt;, &gt;, and).

The start of sections and the various levels of subsections are identified using regular expressions and an XML-tag (e.g., <section>) is inserted in the text. The start of tables, figures and comments are found the same way (captions are above tables and figures in all the documents). Everything before the first section is removed. This is the introduction and the table of contents. Last, the XML preamble and the end document tag are inserted. End tags for each of the different tags (section, subsections, table, figure) are inserted when a new element of the same type starts, a new parent element starts, or where the document ends. Sometimes, tables in the document contain entries that are confused with section numbers or the document contained difficult structuring (e.g., first a table is within a note and then a note is within a table). In such cases, it is necessary to manually correct the XML file.

4.3 Extraction of requirements sentences

We parse the generated XML-file with a DOM parser and extract the textual content of each of the tags that contain requirement text. We do not use content from tables, figures and guidance notes. Each element is split, using the NLTK's recommended sentence tokenizer [6], into individual sentences. Finally, we collect all the sentences that contain the word shall (i.e., it is a mandatory requirement) and write them to a TSV file including the numbering of each of the sentences for reference.

4.4 Creation of labelled data

We use *snorkel* as a framework for creating a weak supervision-based training dataset. The framework is based on providing multiple labelling functions (LFs) that aim to classify a subset of the dataset. The framework unifies the output of the LFs and generates weak supervision labelled data. We have two labels **SCOPE**, **NOT SCOPE**. This is thus a binary document classification task. Each of our label functions works independently and, given a sentence, either classifies it into **SCOPE** or **NOT SCOPE** or abstains from classifying.

4.4.1 LF using dataset related to ISO 15926

Since we limit the approach to pieces of equipment, we use a gazetteer of artifacts. To compile this gazetteer, we used the SPARQL endpoint available at [1], which contains structured data according to the ISO 15926 standard. First, we queried for a list with all labels of concepts that are (recursively) subclasses of the ARTEFACT CLASS (`<http://data.15926.org/rdl/RDS201644>`). The result of this query was curated as follows: *i)* all strings were converted to lowercase, *ii)* the substring "class" was removed from the end of all class-names, *iii)* the substring "asme" was removed from the beginning of all class-names *iv)* if a string ends with "for asme ... ", the substring starting with "for asme" to the end of the label was removed, *v)* duplicates were removed from the list.

Second, we queried for all concepts that are (recursively) subclass of ARTEFACT (`<http://data.15926.org/rdl/RDS422594>`). All strings were converted to lowercase. The two lists of terms were then concatenated into a final list of 10861 terms $\mathcal{T}_{15926}$. The labelling algorithm is described in Algorithm 1.

Algorithm 1 LF using dataset related to ISO 15926.

```
Input: $\mathcal{T}_{15926}, r$
Output: SCOPE or ABSTAIN
1: $\mathcal{C} \leftarrow$ get_noun_chunks($r$)
2: for all $c \in \mathcal{C}$ do
3:     $s \leftarrow$ lemmatize_each_word(remove_stopwords(lowercase(word_tokenize($c$))))
4:     while $s$ not empty do
5:         if $s \in \mathcal{T}_{15926}$ then return SCOPE
6:         else $s \leftarrow$ remove_first_word($s$)
   return ABSTAIN
```

4.4.2 LF using list of words from TermoStat

We used the online TermoStat tool [17, 18] to extract domain vocabulary. We collected nouns both single and multi-word terms. We transformed each PDF-document into text using pdftotext and passed the text-documents one by one and collected all the terms in one list. The list was reduced by removing duplicates to 37,720 terms.

The list of domain vocabulary was filtered using WordNet. All terms that have hyponyms that are in the same synset as "artifact" were kept. This gave us a final list of 974 words, $\mathcal{T}_{term}$. The list includes words we probably wouldn't label as equipment, however, it is quite specific to the domain. Therefore it seemed best to create a function that, instead of label **SCOPE** all sentences that contain the word, it will label **NOT SCOPE** if after checking all chunks in the sentence, nothing was found. The labelling algorithm is described in Algorithm 2.

4.4.3 LF using list of words generated using word vectors

We used pre-trained word embeddings from gensim [3] (glove-wiki-gigaword-100). We manually compiled a small list of words (seed-terms) containing known pieces of equipment. The words were picked from ISO 14224 [22] and a few terms from [12] and [10] in total 30 words. The words are listed in Section A.2.

Algorithm 2 LF using list of words from TermoStat.

```
Input: $\mathcal{T}_{term}, r$
Output: NOT SCOPE or ABSTAIN
 1: $\mathcal{C} \leftarrow$ get_noun_chunks$(r)$
 2: for all $c \in \mathcal{C}$ do
 3:     $s \leftarrow$ lemmatize_each_word(remove_stopwords(lowercase(word_tokenize$(c)$)))
 4:     while $s$ not empty do
 5:         if $s \in \mathcal{T}_{term}$ then
 6:             $found \leftarrow True$
 7:             Break
 8:         else remove_first_word$(s)$
 9: if $found == True$ then return ABSTAIN
10: else return NOT SCOPE
```

First, we calculated the average vector of these pieces of equipment. Then, we gathered the top 100 most similar terms (cosine similarity) to this vector and created a list $\mathcal{T}_{emb}$. For the labelling, we reused Algorithm 1, but with $\mathcal{T}_{emb}$ instead of $\mathcal{T}_{15926}$. We did not do any fine-tuning of the words used as seeds or of the vectors used, so it should be possible to improve upon this function.

4.4.4 LFs using simple regular expression patterns

We defined two label functions using simple regular expression patterns:

- If the sentence contains a colon: **NOT SCOPE** (often a definition)
  "Pressure, Maximum Allowable Incidental (MAIP): In relation to pipelines, this is ..." is **NOT SCOPE**
- If it contains a capitalized "For": **SCOPE**
  "For full-lift safety valves, ..." is **SCOPE**

If the sentence does not match the pattern, **ABSTAIN**.

4.4.5 LFs using simple terms

We also used two functions checking for terms in a sentence:

- Using a small manually created list of terms commonly seen in **SCOPE** sentences – see the Appendix.
- A manually created list of terms commonly seen in **NOT SCOPE** sentences – see the Appendix.

We did the most effort on compiling the list for the **NOT SCOPE** sentences because there is no dictionary that we can use to identify a sentence that does not contain any piece of equipment. The lists were created by manually looking at the requirements. Beginning with a few terms (e.g., report, carried out) we kept looking at the identified **NOT SCOPE**-sentences to identify terms that were common among them.

4.5 The classifier

To build the classifier, we use a pretrained BERT model [16, 4] (bert-base-cased) with a linear layer on top of the pooled output. For training, we use the values proposed in the original paper (and did no further tuning of hyper-parameters). The parameters are shown in Table 1. All parameters of the BERT model except bias, gamma and beta parameters are trained. The Linear scheduler with warmup is used.

Table 1 Hyper-parameters used to train the classifier.

Parameter	Value
epochs	4
learning rate	3e05
eps	1e-8
max seq len	32
optimizer	ADAMW
weight-decay-rate	0.01

5 Evaluation

5.1 Manual labelling

We created a labelling guideline for scope detection describing the task, the limiting conditions and example labelling of confusing cases. We used the guidelines and manually labelled 200 requirements sentences to evaluate performance when developing the dataset and the performance of the final model. When training the classifier, we also used 10% of the dataset for evaluation. Also, we divided the 300 sentences into three Excel sheets and asked ontology experts from DNV GL to annotate these sheets. Three annotators were selected for each sheet and they were given the same annotation guidelines that we had been using.

We further annotated 200 sentences from *Drilling facilities* (DNVGL-OS-E101) [9], all extracted sentences (180) from *Floating docks* (DNVGL-RU-FD) [11], and 200 sentences from Equinor's *Field instrumentation* (TR3032) [13].

5.2 Automatic labelling

The accuracy of the dataset with regard to the labelled dataset is 0.79 using *snorkel*'s majority vote model which was empirically found to give higher accuracy than *snorkel*'s label model (acc 0.76) for the task.

5.3 The model

The created training data was split into train and validation. The result of running the classifier on the validation dataset is found in Table 2. The accuracy of the classifier is 0.91. The result of running the classifier on the 200 manually labelled sentences is shown in Table 3 together with the results from the experiments with the three other documents, DNVGL-OS-E101, DNVGL-RU-FD, and TR3032.

Table 2 Result of the evaluation of the classifier on the validation set.

Class	Precision	Recall	F1	Support
NOT SCOPE	0.86	0.81	0.83	638
SCOPE	0.93	0.95	0.94	1801

[4] This is the document that the classifier was trained on.

Table 3 Evaluation of the classifier on manually labelled data from different documents.

Dataset	Class	Precision	Recall	F1	Support
RU-SHIP and ST-F101[4]	**NOT SCOPE**	0.87	0.64	0.74	92
Acc: 0.79	**SCOPE**	0.75	0.92	0.83	108
OS-E101	**NOT SCOPE**	0.80	0.47	0.60	76
Acc: 0.76	**SCOPE**	0.74	0.93	0.82	124
RU-FD	**NOT SCOPE**	0.60	0.46	0.52	68
Acc: 0.68	**SCOPE**	0.71	0.81	0.76	112
Equinor TR3032	**NOT SCOPE**	0.76	0.52	0.61	66
Acc: 0.79	**SCOPE**	0.79	0.92	0.85	134

5.4 Labelling by DNV GL

The manually labelled data from DNV GL were combined into one dataset using majority vote. Whenever an annotator had left a sentence unlabelled, we labelled it as scope if any of the two others had labelled it as scope. When there were missing values, however, it was not possible to calculate Kohen's kappa. For the second Excel sheet, we got only two of the three annotated sheets, so to combine the two sheets we labelled a sentence as scope if any of the two annotators had labelled it as scope. The accuracy of the classifier on the combined data was 0.67. Precision, recall and F1-score is shown in Table 4. We calculated inter-annotator agreement for each of the Excel sheets and the Kohen's kappa and Krippendorf's alpha scores are presented in Table 5 and Table 6 respectively.

Table 4 Evaluation of the classifier on the dataset labelled by ontology experts acc: 0.67.

Class	Precision	Recall	F1	Support
NOT SCOPE	0.84	0.48	0.61	162
SCOPE	0.59	0.89	0.71	138

Table 5 Kohen's kappa score between annotators.

Sheet no.	Annotators	Kohen's kappa
1	A B	-
	A C	0.335
	B C	-
2	C D	-
3	B E	0.582
	A E	0.368
	A B	0.408

Table 6 Krippendorff's alpha.

Sheet no.	Krippendorff's alpha
1	0.396
2	0.510
3	0.434

5.5 Examples of classification

The requirements sentences in this section are reproduced with permission from DNV GL.[5] Using the colour **olive**, we indicate the scope of the sentence. By the colour **red**, we indicate something that can be seen as a scope but is not considered in this paper.

5.5.1 Examples of correct classifications

- DNV-ST-F101 Appendix B MECHANICAL TESTING AND CORROSION TESTING **B.2.3.9** [sentence 1] ***Test pieces for transverse weld (cross weld) tensile*** *shall be rectangular and in accordance with [B.2.3.3]* **SCOPE**
- DNV-RU-SHIP Pt.5 Ch.13 Sec.13 RADIATION HAZARDS **4.1.2** [sentence 2]*Attention shall be paid to effects from re-radiated fields* **NOT SCOPE**
- DNV-RU-SHIP Pt.7 Ch.1 Sec.24 Winterized vessels **24.2.1** [sentence 1] ***Anti-icing and de-icing switchboards*** *shall be surveyed.* **SCOPE**.
- DNV-RU-SHIP Pt.5 Ch.10 Sec.14 SLOP RECEPTION VESSEL **2.2.4** [sentence 2] *Coamings of suitable height shall be arranged below* ***manifolds*** *and* ***hose connections*** *in order to minimize spill.* **SCOPE**
- DNV-RU-SHIP Pt.5 Ch.10 Sec.6 DIVING SUPPORT VESSELS **2.1.4** *If the diving support vessel does not carry the class notation COMF(V- crn),* ***the diver's accommodation area (inner area)*** *shall be subject to the relevant vibration and noise measurements applicable to the remaining accommodation.* **SCOPE**

5.5.2 Examples of incorrect classifications

In this section, we list some incorrect classifications and provide a short comment.

- DNV-RU-SHIP Pt.6 Ch.2 Sec.13 GAS FUELLED SHIP INSTALLATIONS - GAS FUELLED LPG **9.3.3.2** [sentence 2] ***Detection of leakages*** *shall result in automatic closing of all valves required to isolate the leakage.* Falsely classified as **SCOPE**. Here, we don't know what we are detecting the leakage on. Fails possibly because a piece of equipment occurs in the sentence without being the scope.
- DNV-RU-SHIP Pt.4 Ch.3 Sec.2 GAS TURBINES **2.2.5** [sentence 7] ***Burner*** *lifetime shall be specified together with the nominal/recommended exchange intervals.* Falsely classified as **NOT SCOPE**. It should be classified as **SCOPE** because lifetime as a direct property of burner.
- DNV-RU-SHIP Pt.7 Ch.1 Sec.2 ANNUAL SURVEYS EXTENT – MAIN CLASS **3.3.1** [sentence 1] ***The survey on deck*** *shall include: – examination of the venting systems for the cargo tanks, interbarrier spaces and hold spaces.* Falsely classified as **SCOPE**. It is about the survey which is not a piece of equipment.

[5] © DNV GL. DNV GL does not take responsibility for any consequences arising from the use of this content.

- DNV-RU-SHIP Pt.5 Ch.10 Sec.14 SLOP RECEPTION VESSEL
  3.1.3 [sentence 2] *– **Two-way communication between the reception facility and the delivery vessel** shall be established before the transfer commences.* Falsely classified as **SCOPE**. It is about communication which is not a piece of equipment.
- DNV-RU-SHIP Pt.5 Ch.9 Sec.4 STANDBY VESSELS
  2.2.3 [sentence 1] *The net plate thickness, in mm, in **superstructures** and **deckhouses** shall not be less than: where: t0 = 4.5 for front bulkheads and weather deck forward of the lowest tier of the front bulkhead = 3.5 for sides and aft end bulkheads and weather decks elsewhere = 3.0 for superstructure and deckhouse decks (in way of accommodation) c = coefficient taken as:.* Falsely classified as **NOT SCOPE**.
- DNV-RU-SHIP Pt.6 Ch.1 Sec.4 STRENGTHENED FOR HEAVY CARGO IN BULK - HC
  10.1.2 [sentence 1] *Requirements specific to dry cargo ships The loading manual shall contain the loading conditions described in [4].* Falsely classified as **SCOPE**. It is about the loading manual, but the title seems to have been joined together with the requirement making it look like it is about dry cargo ships.
- DNV-ST-F101 Sec.7 CONSTRUCTION – LINEPIPE
  7.4.2.3 *In addition to the applicable information given in [7.1.7] and [7.1.8], **the MPS for lined linepipe** shall as a minimum contain the following information (as applicable): – details for fabrication of backing pipe and liner – quality control checks for the lining process – details of data to be recorded (e g expansion pressure/force, strain, deformation) – procedure for cut back prior to seal welding or cladding to attach liner to carrier pipe – seal welding procedures – details regarding any CRA clad welding to pipe ends.* Falsely classified as **SCOPE**. It is about MPS (manufacturing procedure specification) which is a document.
- DNV-ST-F101 Sec.6 DESIGN - MATERIALS ENGINEERING
  6.3.5.3 [sentence 2] *For **concrete coating** the minimum requirements in ISO 21809-5 shall apply with additional requirements given in [9.3] in this standard.* Classified as **SCOPE**, we regard concrete coating a material and classify the sentence as **NOT SCOPE**.

5.5.3 Sentence length and incorrect classification

Since our labelling functions identify terms in a sentence, it seems likely that we incorrectly classify longer sentences more often because they contain more terms. However, that this is not the case. The distribution of the incorrectly classified sentences is almost the same as the distribution of the entire dataset when plotted against sentence length as seen in Fig. 2.

6 Discussion

That majority vote gave better results than *snorkel*'s label function is in line with the results in [27] that when we have few labelling functions, many data points will have only one labelling function giving something different from **ABSTAIN**. From Table 7 in Appendix A, it is possible to see that the labelling functions that label **SCOPE** have better coverage than the ones that label **NOT SCOPE** and Table 8 show that the dataset performance is shifted toward higher recall on **SCOPE** and high precision and low recall on **NOT SCOPE**.

The detection of **NOT SCOPE** is in essence the detection of the lack of (technical) scope terms in a sentence. To detect the lack of terms we must have an exhaustive list of terms which we cannot assume to have. Having some terms that indicate **NOT SCOPE** (see the Appendix) is beneficial, but we suspect that many more patterns could be identified. It could be interesting to use clustering or frequent itemset mining to identify such terms.

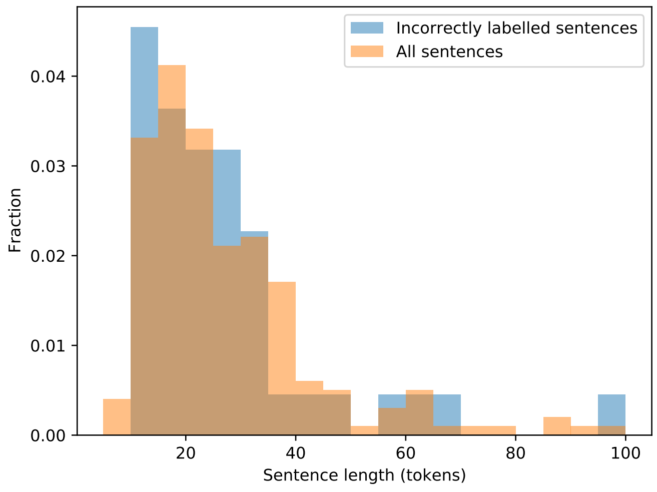

Figure 2 Distribution of the number of sentence per sentence length (number of tokens). The x-axis depicts the number of tokens in a sentence. The y-axis depicts the fraction of the total number of sentences.

For the word vector label function (Section 4.4.3) it would be interesting to try different sets of seed words and different word vectors. Using domain-specific word embeddings can also prove to be interesting.

The accuracy of the classifier is the same as using the dataset, 0.79. Taking a 95% confidence interval of the classifier's accuracy, we get 0.79 ± 0.0565 or $\approx [0.7335, 0.8465]$. The results of the classifier are encouraging. Having a classifier that can identify the relevant sentences for the scope detection task is a step toward detecting the scope of a requirement. As far as we know, the task of scope detection has not been tackled before in the literature.

We can see from the results in Table 3 that, as for the labelling functions, the model is also better at identifying **SCOPE** sentences than identifying **NOT SCOPE**. In general, recall is very high for **SCOPE** but rather low for **NOT SCOPE** sentences.

The evaluation of the approach on other documents shows promising results on both different topics and cross-company. The floating dock document shows the worst performance. This may be due to the vocabulary in this document that is not present in training data or the use of a more complex sentence structure. Somewhat surprisingly, the performance on Equinor's requirements document is comparable to the performance on the dataset from the same document the classifier was trained on.

We find, as expected that the model can falsely classify a sentence as **SCOPE** if the sentence contains equipment terms that are not scope. It does, however, also label many of them correctly. Sentences with quite complex scopes are also often classified correctly. We observe another challenge when the scope is a term that is followed by a non-scope-term as is the case for "burner lifetime". Several of the cases where the model disagreed with the manual labelling were also challenging to label manually and may be open for discussion.

We found no relation between the number of tokens in the sentence and misclassification (see Figure 2). It does, however, still seem that it has a higher probability of misclassification where the sentence contains equipment concepts that are not scopes. Some of the misclassifications are also very difficult for a human annotator. That may be because of the lack of context, ambiguous requirements or lack of domain knowledge.

For the manual labelling by the experts, we observe only moderate agreement among the annotators. We thought to clarify the annotation task by explicitly limiting the types of scopes to pieces of equipment. The boundaries of the equipment class can, however, be somewhat unclear. Thus, annotators did not agree on what is a piece of equipment in all cases. The lack of context information for each sentence also opens up for different interpretations by different annotators e.g., if a sentence says that an alarm shall sound if a certain condition is met, one annotator may consider that the sentence refers to the sound of the alarm and another annotator may consider it to be the physical entity that makes the sound. These results may also open up for discussions around some of the requirements sentences and how to create easier to understand requirements to better ensure a common understanding. To write clear and concise requirements is challenging and is the topic of [7].

Since a substantial amount of the sentences in the dataset does not contain a scope, training a classifier on the entire dataset to detect scopes would give a very imbalanced dataset and in a lot of sentences, it would not find anything. Manual labelling for scope detection would also be easier if most of the sentences actually contain a scope.

7 Conclusion

We have trained a model that classifies a requirement sentence into either **SCOPE** or **NOT SCOPE** depending on if the sentence contains the scope of the requirement or not. By using data programming we label a dataset with about 30.000 requirements sentences and train a simple BERT-based binary classifier on this dataset. The model shows good performance in separating sentences without a scope from sentences with scope with an accuracy of 0.79 on manually labelled sentences from the same document. The model also shows promising results on documents from other related domains and a document from another company. The performance of the model is, however, shifted toward high recall 0.92 for sentences with scope as opposed to a recall of 0.64 for sentences without scope.

By extracting the individual requirements from the document and splitting them into individual sentences, important contextual information is inevitably lost. Still, with this simplifying assumption, this seems like a promising first step toward scope detection of textual requirements. One major challenge for this task is that it is not trivial to ensure that all the involved parts have a common understanding of the problem, the sentences, and the terminology. The moderate inter-annotator agreements result even after having developed a guideline with many example sentences demonstrates this challenge.

References

1 15926browser. (visited on 2021-02-22). URL: `http://data.15926.org/rdl`.
2 Apache PDFBox | A Java PDF Library. `https://pdfbox.apache.org/` (visited on 2021-12-21).
3 Gensim. `https://radimrehurek.com/gensim/` (visited on 2021-02-17).
4 Google-Research/Bert. `https://github.com/google-research/bert` (visited on 2021-01-27).
5 Iso15926 equipment class. `http://data.15926.org/rdl/RDS8615020` (visited on 2021-02-22).
6 Natural Language Toolkit – NLTK. (visited on 2021-02-08). URL: `https://www.nltk.org/`.
7 INCOSE – guide for writing requirements, 2017.
8 S. Abualhaija, C Arora, et al. A Machine Learning-Based Approach for Demarcating Requirements in Textual Specifications. In *RE 2019*, pages 51–62, 2019.
9 Det Norske Veritas AS. Drilling facilities. Technical report, DNV-OS-E101, Ed. January 2018. © DNV GL.
10 Det Norske Veritas AS. Rules for classification: Ships. Technical report, DNV-RU-SHIP, Ed. July 2019. © DNV GL.

11 Det Norske Veritas AS. Floating docks. Technical report, DNVGL-RU-FD, Ed. October 2015. © DNV GL.

12 Det Norske Veritas AS. Submarine pipeline systems. Technical report, DNV-OS-F101, Ed. October 2017. © DNV GL.

13 Equinor ASA. Field instrumentation. Technical report, TR3032, Ver 3, August 2011. ©Equinor.

14 H. Bast and C. Korzen. A Benchmark and Evaluation for Text Extraction from PDF. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 1–10, 2017.

15 Agustin Casamayor, Daniela Godoy, and Marcelo Campo. Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Information and Software Technology*, 52(4):436–445, 2010.

16 Jacob Devlin, Ming-Wei Chang, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv*, 2019. `arXiv:1810.04805`.

17 Patric Drouin. TermoStat Web. `http://termostat.ling.umontreal.ca/` (visited on 2021-02-17).

18 Patrick Drouin. Term extraction using non-technical corpora as a point of leverage. *Terminology*, 9:99–115, 2003.

19 G. Fantoni, E. Coli, et al. Text mining tool for translating terms of contract into technical specifications: Development and application in the railway sector. *Computers in Industry*, 124:103357, 2021.

20 Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. *arXiv*, 2018. `arXiv:1801.06146`.

21 IBM. DOORS. `https://www.ibm.com/uk-en/products/requirements-management` (visited on 2021-03-08).

22 BS ISO. Iso 14224,“petroleum and natural gas industries: collection and exchange of reliability and maintenance data for equipment“. *British Standards Institution, UK*, 1999.

23 Menon Economics. Requirements as cost drivers in the Norwegian petroleum industry. `https://www.menon.no/requirements-as-cost-drivers-in-the-norwegian-petroleum-industry` (visited on 2021-02-19).

24 Mike Mintz, Steven Bills, et al. Distant supervision for relation extraction without labeled data. In *ACL/AFNLP*, volume 2, pages 1003–1011, 2009.

25 Farhad Nooralahzadeh, Jan Tore Lønning, and Lilja Øvrelid. Reinforcement-based denoising of distantly supervised ner with partial annotation. In *DeepLo Workshop*, 2019.

26 Farhad Nooralahzadeh, Lilja Øvrelid, and Jan Tore Lønning. Evaluation of domain-specific word embeddings using knowledge resources. In *LREC 2018*, 2018.

27 Alexander Ratner, Stephen H Bach, et al. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3), 2017.

28 Benedetta Rosadini, Alessio Ferrari, et al. Using NLP to detect requirements defects: An industrial experience in the railway domain. In *REFSQ*, pages 344–360, 2017.

29 Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *NAACL Tutorials*, pages 15–18, 2019.

30 SIEMENS. Polarion REQUIREMENTS. `https://polarion.plm.automation.siemens.com/products/polarion-requirements` (visited on 2021-03-08).

31 SIRIUS. DREAM and READI: Cooperation to Manage Digital Requirements. `https://sirius-labs.no/dream-and-readi-cooperation-to-manage-digital-requirements/` (visited on 2021-03-08).

32 SIRIUS and DNV GL. On the READI method. Personal communication.

33 Jonas Winkler and Andreas Vogelsang. Automatic Classification of Requirements Based on Convolutional Neural Networks. In *RE Workshops*, pages 39–45, 2016.

A Additional details about the labelling functions

A.1 Coverage, overlap and performance

The coverage, overlap and conflicts among the labelling functions is presented in Table 7. We also include in Table 8 the performance of the dataset on the manually labelled sentences. It is important to notice, however, that these metrics are not directly comparable to the performance of the model as only considers sentences that are classified and ignores the **ABSTAIN** labels.

Table 7 Coverage, overlap and conflicts of the labelling functions.

Labelling function	Polarity	Coverage	Overlaps	Conflicts
has_equipment_termostat	**NOT SCOPE**	0.119	0.057	0.016
has_equipment_iso15926	**SCOPE**	0.523	0.355	0.138
has_equipment_wv	**SCOPE**	0.384	0.307	0.100
contains_colon	**NOT SCOPE**	0.105	0.094	0.082
contains_For	**SCOPE**	0.073	0.064	0.033
contains_scope_words	**SCOPE**	0.039	0.033	0.009
contains_non_scope_words	**NOT SCOPE**	0.215	0.170	0.124

Table 8 Precision, recall, f-score and support for the dataset on the manually labelled sentences disregarding all **ABSTAIN** labels (thus not comparable to the performance of the model).

Class	Precision	Recall	F1-score	Support
NOT SCOPE	0.87	0.65	0.74	71
SCOPE	0.79	0.93	0.85	100

A.2 List of seed words used for generating the word vector

The list of words used for generating the average word vector in 4.4.3: {“riser”, “accumulator”, “engine”, “compressor”, “blower”, “controller”, “generator”, “turbine”, “pipeline”, “sensor”, “pump”, “vessel”, “valve”, “bolt”, “cable”, “clamp”, “connector”, “cooler”, “fan”, “filter”, “fitting”, “flange”, “gearbox”, “joint”, “pipe”, “nut”, “pump”, “reflector”, “tube”, “ship”}

A.3 List of terms used for the LFs using simple terms

The words used for identifying **SCOPE** in Section 4.4.5: {“shall be capable of”, “shall be designed”, “shall be tested”}

The words used for identifying **NOT SCOPE** in Section 4.4.5: {“report”, “survey”, “shall include”, “describe”, “description”, “drawing”, “parameters”, “parameter”, “results”, “examination”, “include”, “include:”, “shall be taken as”, “shall be taken as:”, “carried out”, “shall be used to”, “shall cover”, “be based on”, “be performed”, “evaluate”, “calculation”, “calculations”, “analysis shall”, “criteria”, “based upon”, “determined”, “details’, “references”, “inspection”, “testing shall”, “hence”, “procedure”, “procedures”, “review”, “testing”}