Extracting Software Requirements from Unstructured Documents

Vladimir Ivanov[1(✉)], Andrey Sadovykh[1,2], Alexandr Naumchev[1], Alessandra Bagnato[2], and Kirill Yakovlev[1]

[1] Innopolis University, Innopolis, Russia
v.ivanov@innopolis.ru
[2] Softeam, Paris, France
http://university.innopolis.ru, http://www.softeam.com

Abstract. Requirements identification in textual documents or extraction is a tedious and error prone task that many researchers suggest automating. We manually annotated the PURE dataset and thus created a new one containing both requirements and non-requirements. Using this dataset, we fine-tuned the BERT model and compare the results with several baselines such as fastText and ELMo. In order to evaluate the model on semantically more complex documents we compare the PURE dataset results with experiments on Request For Information (RFI) documents. The RFIs often include software requirements, but in a less standardized way. The fine-tuned BERT showed promising results on PURE dataset on the binary sentence classification task. Comparing with previous and recent studies dealing with constrained inputs, our approach demonstrates high performance in terms of precision and recall metrics, while being agnostic to the unstructured textual input.

Keywords: Software requirements · Requirements elicitation · BERT · FastText · ELMo · Sentence classification

1 Introduction

Identifying and understanding technical requirements is a challenging task for many reasons. Requirements come from various sources and in various forms. These different forms may serve different purposes and levels, from initial project proposals down to detailed contracts of individual methods. The forms that appear earlier in the software process are consumed by more business-oriented stakeholders. Technical people, such as testers, require detailed specifications produced later in the software process to convert them into tests. The evolving form of describing the initial problem is natural and reflects the process of understanding the problem better with time. The available requirement datasets reflect those variations.

One of such is the PURE dataset [1] presented by A. Ferrari et al., which includes 79 Software Requirements Specification (SRS) documents. Mainly this

set is based on the Web search and includes documents with different peculiarities as well as a lexicon with the widespread writing style of requirements. Authors argue that this dataset fairly fits for various Natural Language Processing (NLP) tasks such as requirements categorisation, ambiguity detection, and equivalent requirements identification, etc. But at the same time, additional expertise, as well as an annotation process, are primarily required. However, some other datasets might be challenging. The Request for Information (RFI) documents in itself specify an initial set of "best wishes" for a required solution as well as a variety of constraints in terms of deployment, compatibility or guarantees for maintenance and support. Those documents are meant for engineers though the language used, format, styles and structure are far from being standard. Those parameters largely depend on the organization that issued those documents, personnel involved in designing specifications and traditional practices of the domain the organization belongs to. On the other side, the provider of the solution must quickly qualify the RFI and prepare a well-thought offer that would maximize the value for the customer at minimum costs for the provider. Such analysis is mostly done manually that is costly, time-consuming and error-prone.

In this paper, we present an approach for extraction of requirements coming from unstructured text sources. We fine-tuned the Bidirectional Encoder Representations from Transformers (BERT) [2] model with PURE-based manually annotated dataset of 7,745 sentences to identify technical requirements in natural text. The ReqExp showed 92% precision and 80% recall on the PURE dataset. In order to evaluate BERT, we compared the results with previous state-or-art methods such as fastText and Embeddings from Language Model (ELMo) with an SVM classifier on the same PURE-based dataset, where BERT showed superior results in the precision, but worse results in the recall. Afterwards we carried-out additional experiments with more semantically difficult texts from RFIs. We applied all the candidate models trained on PURE-based dataset to a private dataset of 380 sentences extracted from RFI documents in order to evaluate the possibility to transfer the result to a new domain. BERT again showed superior results in precision (90%), but inferior in recall (71%). With a superior recall metrics fastText (82%) and ELMo (72%) baseline methods showed less false negatives in detecting requirements, i.e. passed through less of undetected requirements. The results obtained in the limited set of experiments showed optimistic outcomes. We ran an additional experiment with a larger training set, consisted of all the text samples from the PURE and obtained precision of 86% and recall of 84% that suggests further fine-tuning of ReqExp on a larger dataset. Moreover, the analysis of related work indicated that the ReqExp approach has certain advantages, since, for example, it is agnostic to the type and the structure of input documents.

Unfortunately, the datasets of comparable studies are not available for a thorough analysis. In order to avoid the same problem with our study, we provide our dataset with manually annotated sentences from the PURE corpus to the community for comparison studies. The initial intention for this study came

from to Softeam company from France. They kindly provided the dataset of RFI documents and endorsed our work. Softeam colleagues confirmed that the results are promising for the industry with a high potential to be used in the company's products and processes.

To summarize this paper, the contributions of this work are: (1) adapted PURE-based dataset for classification tasks with manually annotated sentences containing both requirements and non-requirements; (2) the description of BERT-based tool chain applying NLP for requirements extraction; (3) the results of evaluation on SRS sentences from the PURE dataset in comparison with baseline methods – fastText and ELMo; (4) the results of evaluation on a provide dataset of RFI documents; (5) analysis of comparable studies. Therefore, the current paper starts with the outline of the ReqExp approach in Sect. 2 describing the conceptual method and the dataset construction for profiling BERT neural network, followed by classifier models details in Sect. 3 , continued by validation of the prototype on PURE-based dataset and real RFI documents in Sect. 4, to be concluded by a comparison with the related work in Sect. 5.

2   Methods and Datasets

In this section we describe the process of building a binary sentence classifier. This process has several steps presented in Fig. 1. The ultimate goal was to efficiently build a training dataset that can be used to fine-tune the BERT model and compare it with other advanced NLP methods. Bidirectional Encoder Representations from Transformers (BERT) is a model by Google pre-trained on a huge text corpus [2]. BERT was tested on multiple NLP tasks and achieved state-of-the-art performance. A common way to use BERT for a domain-specific task, such as software requirements extraction, is to fine-tune the model i.e., to optimize model's weights on a domain-specific dataset. To solve a binary classification problem, such dataset should contain two subsets of sentences: (i) a subset of sentences that contain requirements, and (ii) a subset of sentences that does not contain requirements. However, there were no such dataset readily available for requirements extraction.

Thus, the first step of our approach is creating the domain-specific training dataset for fine-tuning. To speed up the process of corpus creation, we decided to process an existing corpus of software specification documents. In [1], Ferrari et al. presented a text corpus of public requirements documents (PURE)[1] The dataset contains 79 publicly available natural language requirements documents. We applied PURE corpus as our main source of both requirements and non-requirements for designing our own dataset. Initially, documents were presented in the form of raw text with different format, structure and writing style.

At the beginning we made an attempt to extract sentences with requirements using some parsing and extraction engines. Unfortunately, this idea was rejected due to unsatisfactory quality of extracted text samples. Despite that authors did provide a set of text requirements in the form of *.json* files to simplify the

---

[1] Data of the corpus can be found here: https://zenodo.org/record/1414117.

extraction process, this was not enough to design a representative subset for each class. It was decided to process documents manually preserving initial syntax and semantics in the extracted text samples. Still this process was associated with some challenges, which required us to apply the following conventions:

- Due to different length and structure of requirements, it was decided to assume one sentence as one data unit. It means that a requirement, which consists of several sentences, was divided into several units.
- Sometimes requirements were written in the form of a list with many elements. In such case, this list was transformed into one sentence separating parts by commas. In some rare cases, where elements were individual sentences, such requirement was divided into several sentences accordingly.

Overall, we extracted 7,745 sentences, where 4,145 were requirements and 3,600 were non-requirements from 30 documents. The extraction was done from the appropriate sections of the documents. Usually each document provides structural elements like table of elements, requirements-focused sections with appropriate names or contextual footnotes. The latter one might have some specific numbering used in a document and initially explained by the authors, or lexical elements that are usually inherent for requirements such as modal verbs like 'must', 'shall', 'should', etc. When identifying requirements and non-requirements we followed principle described in [3].

Sentences with requirements were manually extracted by one specialist in Software Engineering area and then independently validated by another expert from our research group. Some special cases were separately discussed in the group to make annotation more consistent. In the case when there was no agreement concerning which label should be utilized to those text samples, they were removed to maintain consistency of the set.
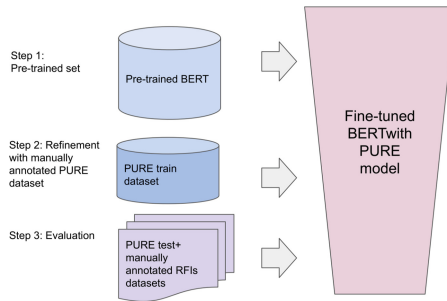

Fig. 1. The training process to obtain a model for binary sentence classification for requirements extraction.

3   Models for Sentence Classification

For this study we chose two baseline approaches, which preceded the BERT architecture, for binary sentence classification. Specifically we considered fast-Text model and ELMo embeddings with a classical machine learning classifier. The latter one was chosen through an experiment stage to choose the most promising model based on the most explicit and informative metrics during comparison.

3.1   FastText Baseline Model

fastText is a method for constructing word embeddings based on the Word2Vec principle, but with some extension. An embedding for each word is learnt from embedding vectors for character n-grams that constitute the word. Thus, each word vector is represented as the sum of the n-gram vectors. This extends the Word2Vec type models with subword information. Once a word is represented using character n-grams, a skipgram model is trained to learn the embeddings. This approach provides a fast and scalable solution, allowing to train models on large corpora [4]. For the down-stream task (binary classification of sentences) we use a linear classifier that is default (a multinomial logistic regression) in the fastText library.

An important part in designing predictive models is associated with finding the best hyper-parameters. Usually, it is a cumbersome and time-consuming task, especially when doing it manually. For this purpose, we applied an auto-tune feature located inside the fastText functionality by using the validation set presented earlier[2].

3.2   ELMo with SVM Classifier

Embeddings from Language Models, or ELMo is an advanced method for constructing a contextualized word representation. The core ability is to model different characteristics of word use together with linguistic context. Specifically, these representations are internal states of a deep bidirectional language model that is pre-trained on a large corpus. ELMo maximizes the log-likelihood of the forward and backward directions in a sequence of words [5]. These produced vectors can be applied as features for various NLP purposes.

To make a decision for distinguishing among sentences with requirement and without, one needs to use a classifier on top of the features extracted from text. To find the best classifier, we compared several popular machine learning models such as Logistic Regression, Random Forest classifier, Support Vector Machines and Multi-Layer Perceptron. To this end the Grid Search was applied with specific set of initially predefined hyperparameters for each model. The process of assessment was based entirely on the training set. As a result we chose with Support Vector Machines with polynomial kernel that showed the

---

[2] https://fasttext.cc/docs/en/autotune.html.

highest result in terms of F1-score. The results related to this baseline will be marked as "ELMo+SVM" in the following sections.

3.3   BERT-Based Approach

Recently using a large pre-trained language model has shown a breakthrough in many tasks of natural language processing including text classification. The baseline approach to transfer learning typically implies pre-training a neural network on a large corpus of texts with further fine-tuning the model on a specific task. In this study, we use pre-trained BERT [2] model[3] as a baseline. In all experiments with BERT, we adopted the implementation of fine-tuning process for binary text classification presented in [6]. Hyper-parameters used in fine-tuning are the following: batch size is 64, learning rate is $2 \cdot 10^{-5}$, max sequence length is 100, number of epochs is 2. Overall schema is presented in Fig. 1.

4   Experimental Setup and Validation

4.1   Experimental Setup

Empirical evaluation of model performance was carried out in two phases. First, we used a classical approach with splitting dataset into disjoint sets of sentences (namely, 'train', 'test' and 'validation' sets). All models were trained on the same 'train' subset and tested on the same 'test' part from the PURE dataset. We did not apply cross-validation due to the special rule of splitting collection into train and test sets, i.e. *sentences from some document should appear in either the train set or in the test set*. The dataset as well as the split is available for the sake of reproducibility and comparison of results.

Second, we used an out-of-sample data for evaluation of the classifiers. To this end we tested the performance on sentences manually extracted from the RFI documents. All models were trained on the same 'train' part and tested on the same group of sentences extracted from RFI. We argue that such experimental conditions are more relevant in practice. Indeed, we expected decreasing of the performance on the RFI documents. This shows that fine-tuning of large pre-trained language models is not a panacea in practice, however it gives a viable alternative to rule-based tools. Finally, we show that adding more data on the fine-tuning phase leads to better performance on the out-of-sample data.

4.2   Train, Test and Validation Sets

As for the data split, we had 7,745 sentences extracted from the PURE dataset to split them up into train, test and validation subsets by the parts of approximately 70%, 20 % and 10% accordingly. In order to consistently separate our

---

[3] Specifically, we use the BERT-base uncased model from https://github.com/google-research/bert.

dataset, sentences from every document were fully added only to one specific subset. We found a particular combination of documents for each subset that preserve the stated separation and at the same time preserves the balance of class labels. Statistics of the split is in sentences is the following: train set (5306); test set (1534); validation set (905). The dataset is available at https://zenodo.org/record/4630687.

4.3   Experiments with PURE Documents

The main purpose of this study was about assessing three major classification metrics: Precision, Recall, and F1-score. Those metrics are perceived as the golden standard in Machine Learning and Deep Learning areas.

Result of the first phase of evaluation is presented in Table 1. As it was expected, more advanced model (BERT) showed better results in terms of F1-score. BERT-based model showed high precision (0.92) and lower recall (0.8). Results of the BERT model is available at https://bit.ly/3oPElMm. However, the values of precision and recall metrics behave differently for the fastText and ELMo-based baselines. fastText-based classifier showed better Recall (0.93) comparing with other architectures. This property might be useful in some cases when it is necessary to extract more relevant sentences and text patterns associated with requirements.

Table 1. Results of experiments with PURE

Model	F1	P	R	TP	TN	FP	FN
fastText	.81	.72	**.93**	763	419	295	57
ELMo+SVM	.83	.78	.88	827	364	231	112
BERT	**.86**	.92	.80	841	407	69	217

4.4   Experiments with RFI Documents

In order to further evaluate our approach on less standardized requirements documents, we applied our prototypes to five anonymized documents provided by Softeam, France, as issued by their customers. We parsed the obtained documents to extract the paragraphs and then triggered the automatic requirements extraction from those paragraphs. After that we manually annotated all the sentences and thus assessed the four major metrics - the number of the true positives, the true negatives, the false positives and the false negatives. Then the other metrics such as precision, recall and F1-score were derived. The results can be found in Table 2.

To the validity of the results, it should be noted that the annotation was done by one single specialist who was aware of the context of the Request for Information (RFI) documents. This choice is justified to ensure the uniformity

of the manual annotation, since we noticed that the understanding of whether a sentence is a requirement depends on context very much. In addition, it should be mentioned that our attempts of manual annotation by junior researchers were unfruitful since the results were completely unsatisfactory.

Table 2. Results of experiments with RFI

Model	F1	P	R	TP	TN	FP	FN
fastText	.65	.54	**.82**	146	83	120	31
ELMo+SVM	.69	.66	.73	177	48	89	66
BERT	**.80**	.90	.71	190	93	21	76

We also faced several challenges when were manually annotating the sentences from RFIs. First, RFI documents often specify technical requirements in the form of a question: "Will your solution provide a particular feature?" These questions are difficult to qualify as technical requirements in traditional sense. Second, a large part of RFI documents concern the constraints for the submission of the response such as delay, length, and language of the response. Should that be qualified as a requirement to documentation? All these issues are postponed to the follow up studies where we intend to refine the results. In particular, the study triggered more reflection about the nature of the requirements, since the specification practice is far from the form specified in the traditional requirements engineering standards.

Finally, we ran an additional experiment when we fine-tuned the best model on the sentences from train, validation and test sets altogether with further evaluation on the sentences from RFI documents. All parameters of fine-tuning were the same as in the first set of experiments. The performance has grown significantly (P = 0.86, R = 0.84, F1 = 0.85) compared to the initial setup with the fine-tuning on the train set only (Table 2). Thus, this indicates a possibility that the quality of the result can be improved further by improving and augmenting train dataset.

5   Related Work

The related work analysis in the present section relies on a recent mapping study of natural language processing (NLP) techniques in requirements engineering [7]. The research group analyzed 404 primary studies relevant to the NLP for requirements engineering (NLP4RE) domain. They mapped these studies into several dimensions and identified the key categories along each dimension. Here, we focus on the *Research Facet* that enumerates categories of research and of evaluation methods. Our contribution falls into the *solution proposal* category: we develop a new solution (a tool) for identifying likely requirements in software-related texts. We used the following methods for evaluating our proposed solution:

– *Experience Report*: "the result has been used on real examples, but not in the form of case studies or controlled experiments, the evidence of its use is collected informally or formally."
– *Laboratory Experiment with Software Subjects (LESS)*: "a laboratory experiment to compare the performance of newly proposed system with other existing systems."

Below we list studies from [7] that fall into the same categories as the ReqExp approach does. Some of the studies make strong assumptions about the respective input and/or output artifacts. We do not compare ReqExp with these studies because we do not make such assumptions: ReqExp takes as input arbitrary technical documents and identifies in them statements that look like potential system requirements. The approaches that work with documents from later stages of the requirements process are:

– ARSENAL [8] deals with detailed behavior specifications taking the form of "shall" statements and automatically converts them into formal models.
– AutoAnnotator [9] assumes to receive requirement documents as input and assigns semantic tags to these documents.
– CHOReOS [10] works with well-formed requirements specified by domain experts.
– Doc2Spec [11] works with JavaDoc and other in-code documentation.
– ELICA [12] extracts and classifies requirements-relevant information based on domain repositories and either existing requirement documents or elicitation transcripts.
– FENL [13] extracts software features from online software reviews.
– GUEST [14] works with well-formed requirement documents to extract goal and use case models.
– GaiusT [15] applies to regulatory documents and extracts statements of specific types with predefined structures.
– NLP-KAOS [16] analyzes research abstracts and produces KAOS goal diagrams from them.
– SAFE [17] analyzes descriptions and user reviews of mobile apps in application stores and extracts features from them.
– SNACC [18] automatically checks design information for compliance with regulatory documents.
– Text2Policy [19] processes requirements in natural language to search statements of a predefined type (access control policies).
– UCTD [20] takes use cases as input and produces use cases enriched with transaction trees.
– Guidance Tool [21] works with texts that already look much like use cases and transforms them into properly structured use cases.
– Text2UseCase [22] works with texts that already describe scenarios and transforms them into well-formed use cases.

The recent study [3] has explored the very related topic of demarcating requirements in textual specifications with ML-based approaches. The authors

applied similar methods on a private dataset of 30 specification from industrial companies with presumably 2,145 sentences containing 663 requirements and 1,482 non-requirements candidates. They obtained comparable average precision of 81.2% and an average recall of 95.7%. Unfortunately, it impossible to compare our results on the same dataset. However, we should underline the major differences such as (1) our publicly available dataset is much larger and more balanced; (2) we trained and validated the model on completely different documents; (3) we applied much simpler end-to-end training method without any costly feature engineering; (4) that way our method is arguably more suitable for documents with complex semantics such as RFIs.

Another similar study [23] has presented an application of BERT for multi-class classification of requirements. Authors were focused on the following subtasks:

– Binary classification of functional (F) and non-functional (NFR) requirements on the original dataset.
– Binary and multiclass classification of the four most frequent NFR subclasses on all NFR in the original dataset.
– Multiclass classification of all NFR subclasses on NFR in the original NFR dataset.
– Binary classification of requirements based on functional and quality aspects using the relabeled NFR dataset by another study.

For each subtask they provided classification results in terms of precision, recall and F1-score. As it was expected, BERT showed the most promising results comparing with other baseline approaches.

6   Conclusions

Addressing the customers' needs involves a huge number of documents expressing requirements in various forms. In the initial steps of those interactions with customer the requirements are provided in RFI documents. Those documents often combine technical and commercial requirements as well as provide other constraints. The formats and styles of RFI vary a lot, which makes it difficult to technically qualify the needs and timely prepare an offer.

In this paper we propose a requirements extraction approach based on Machine Learning. We used BERT as the basis of the ML model, which we fine-tuned with a manually annotated dataset constructed from an existing requirements specification repository. We have compared the model to two competitive baselines (fastText-based and ELMo-based). Indeed, the study is preliminary as we just compare baselines models and do not propose any specific improvements for the architectures. However, the results of the baselines are quite promising and can be used in industry.

We validated our approach on five documents coming from Softeam company. The approach and validation results were compared to our previous studies. The five documents were annotated automatically at the sentence level (in total, 380 sentences with 6,676 tokens were extracted) and then evaluated by an expert to calculate the performance scores. This analysis showed that our approach superior, since it supports unconstrained textual specifications - it is agnostic to the format of the input textual document. The performance characteristics of our approach are on the comparable level even without costly feature engineering though a progress can be made. We may further evolve the prototype if, for example, we continue the fine-tuning. In addition, by combining train, test and validation sets into one training set has shown a dramatic increase of BERT performance on more semantically complex RFI documents (P = 0.86, R = 0.84, F1 = 0.85). This suggests usefulness of further work on manual annotation of the remaining documents from the PURE dataset and augmenting the train set.

The collaboration with Softeam made us realize the industrial usefulness of the approach in real cases as it can improve current tools and service by that company. Overall, the obtained results motivate us to continue further evolve our approach. In fact, the improvement will include (i) using a corpus of software-related texts for further pre-training, (ii) fine-tuning models on a multi-task datasets and (iii) annotating more data with requirements.

References

1. Ferrari, A., Spagnolo, G.O., Gnesi, S.: Pure: A dataset of public requirements documents, In: 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 502–505 (2017)
2. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding (2018). arXiv preprint arXiv:1810.04805
3. Abualhaija, S., Arora, C., Sabetzadeh, M., Briand, L.C., Vaz, E.: A machine learning-based approach for demarcating requirements in textual specifications. In: 2019 IEEE 27th International Requirements Engineering Conference (RE), pp. 51–62 (2019)
4. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
5. Peters, M.E., et al.: Deep contextualized word representations (2018). arXiv preprint arXiv:1802.05365
6. Sun, C., Qiu, X., Xu, Y., Huang, X.: How to fine-tune BERT for text classification? (2020)
7. Zhao, L., et al.: Natural language processing (NLP) for requirements engineering: A systematic mapping study (2020). arxiv.org/abs/2004.01099
8. Ghosh, S., Elenius, D., Li, W., Lincoln, P., Shankar, N., Steiner, W.: Arsenal: Automatic requirements specification extraction from natural language. In: Rayadurgam, S., Tkachuk, O. (eds.) NASA Formal Methods, pp. 41–46. Springer International Publishing, Cham (2016)

9. Körner, S.J., Landhäußer, M.: Semantic enriching of natural language texts with automatic thematic role annotation. In: Hopfe, C.J., Rezgui, Y., Métais, E., Preece, A., Li, H. (eds.) NLDB 2010. LNCS, vol. 6177, pp. 92–99. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13881-2_9

10. Maiden, N., Lockerbie, J., Zachos, K., Bertolino, A., De Angelis, G., Lonetti, F.: A requirements-led approach for specifying qos-aware service choreographies: An experience report. In: Salinesi, C., van de Weerd, I. (eds.) Requirements Engineering: Foundation for Software Quality, pp. 239–253. Springer International Publishing, Cham (2014)

11. Zhong, H., Zhang, L., Xie, T., Mei, H.: Inferring resource specifications from natural language api documentation, In: IEEE/ACM International Conference on Automated Software Engineering, pp. 307–318 (2009)

12. Shakeri Hossein Abad, Z., Gervasi, V., Zowghi, D., Barker, K.: Elica: An automated tool for dynamic extraction of requirements relevant information. In: 2018 5th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), pp. 8–14 (2018)

13. Bakar, N.H., Kasirun, Z.M., Salleh, N., Jalab, H.A.: Extracting features from online software reviews to aid requirements reuse. In: Applied Soft Computing, vol. 49, pp. 1297–1315 (2016). www.sciencedirect.com/science/article/pii/S1568494616303830

14. Nguyen, T.H., Grundy, J., Almorsy, M.: Rule-based extraction of goal-use case models from text. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ser. ESEC/FSE 2015. New York, NY, USA: Association for Computing Machinery, pp. 591–601 (2015). https://doi.org/10.1145/2786805.2786876

15. Zeni, N., Kiyavitskaya, N., Mich, L., Cordy, J.R., Mylopoulos, J.: Gaiust: supporting the extraction of rights and obligations for regulatory compliance. Requir. Eng. **20**(1), 1–22 (2015)

16. Casagrande, E., Woldeamlak, S., Woon, W.L., Zeineldin, H.H., Svetinovic, D.: NLP-KAOS for systems goal elicitation: smart metering system case study. IEEE Trans. Softw. Eng. **40**(10), 941–956 (2014)

17. Johann, T., Stanik, C., Alizadeh B, A.M., Maalej, W.: Safe: a simple approach for feature extraction from app descriptions and app reviews. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 21–30 (2017)

18. Zhang, J., El-Gohary, N.M.: Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. In: Automation in Construction, vol. 73, pp. 45–57 (2017). www.sciencedirect.com/science/article/pii/S0926580516301819

19. Xiao, X., Paradkar, A., Thummalapenta, S., Xie, T.: Automated extraction of security policies from natural-language software documents. In: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, ser. FSE '12. New York, NY, USA: Association for Computing Machinery (2012). https://doi.org/10.1145/2393596.2393608

20. Ochodek, M., Nawrocki, J.: Automatic transactions identification in use cases. In: Meyer, B., Nawrocki, J..R.., Walter, B. (eds.) CEE-SET 2007. LNCS, vol. 5082, pp. 55–68. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85279-7_5

21. Rolland, C., Achour, C.B.: Guiding the construction of textual use case specifications," Data & Knowledge Engineering, vol. 25, no. 1, pp. 125–160 (1998). www.sciencedirect.com/science/article/pii/S0169023X97862234

22. Tiwari, S., Ameta, D., Banerjee, A.: An approach to identify use case scenarios from textual requirements specification. In: Proceedings of the 12th Innovations on Software Engineering Conference (Formerly Known as India Software Engineering Conference), Ser. ISEC'19. New York, NY, USA. Association for Computing Machinery (2019). https://doi.org/10.1145/3299771.3299774

23. Hey, T., Keim, J., Koziolek, A., Tichy, W.F.: Norbert: transfer learning for requirements classification. In: 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 169–179 (2020)