COREQQA: A COmpliance REQuirements Understanding using Question Answering Tool

Sallam Abualhaija
sallam.abualhaija@uni.lu
University of Luxembourg
Luxembourg

Chetan Arora
chetan.arora@deakin.edu.au
Deakin University
Australia

Lionel C. Briand
lionel.briand@uni.lu
University of Luxembourg
Luxembourg
& University of Ottawa
Canada

ABSTRACT

We introduce COREQQA, a tool for assisting requirements engineers in acquiring a better understanding of compliance requirements by means of automated Question Answering. Extracting compliance-related requirements by manually navigating through a legal document is both time-consuming and error-prone. COREQQA enables requirements engineers to pose questions in natural language about a compliance-related topic given some legal document, e.g., asking about *data breach*. The tool then automatically navigates through the legal document and returns to the requirements engineer a list of text passages containing the possible answers to the input question. For better readability, the tool also highlights the likely answers in these passages. The engineer can then use this output for specifying compliance requirements. COREQQA is developed using advanced large-scale language models from BERT's family. COREQQA has been evaluated on four legal documents. The results of this evaluation are briefly presented in the paper. The tool is publicly available on Zenodo (https://doi.org/10.5281/zenodo.6653514).

CCS CONCEPTS

• **Software and its engineering** → **Requirements analysis**; • **Computing methodologies** → **Information extraction**.

KEYWORDS

Requirements Engineering (RE), Regulatory Compliance, Natural Language Processing (NLP), Question Answering, Language Models (LMs), BERT.

ACM Reference Format:
Sallam Abualhaija, Chetan Arora, and Lionel C. Briand. 2022. COREQQA: A COmpliance REQuirements Understanding using Question Answering Tool. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3540250.3558926

ESEC/FSE '22, November 14–18, 2022, Singapore, Singapore

ACM ISBN 978-1-4503-9413-0/22/11.
https://doi.org/10.1145/3540250.3558926

1 INTRODUCTION

With the growing reliance on personal data and confidential information, software systems are increasingly subject to compliance against regulations to enforce necessary safeguards for information protection and human safety [13, 16]. Failing to comply with relevant regulations can lead to legal, fiscal or reputational implications for an organization. Regulatory compliance is regarded as an essential yet challenging task by the Requirements Engineering (RE) community [5, 18, 21].

In this paper, we propose the tool COREQQA (**Co**mpliance **Req**uirements Understanding using **Q**uestion **A**nswering). COREQQA is motivated by actual practical needs, considering that requirements engineers are being increasingly involved in software compliance against relevant regulations, e.g., all software systems in Europe must comply with GDPR (Regulation (EU) 2016/679) – the European regulation on data protection, privacy and personal data transfer [11]. Manually handling compliance requirements is tedious and error-prone since requirements engineers have to read through entire legal documents. Such documents are usually hefty, contain complicated natural language (NL) structures, frequently refer to external regulations, and are not easy to peruse without legal expertise [2, 20].

An automated Question Answering (QA) tool such as COREQQA helps requirements engineers efficiently navigate through the compliance-related content of legal documents. QA is the task of automatically finding the answer to a question posed in NL from a given text passage. In our work, we refer to a single text passage as a *context span*. Instead of reviewing long, complex legal documents, COREQQA enables requirements engineers to ask a question about a compliance-related topic, and then returns a list of relevant context spans in which the answer is likely to be found. This way, COREQQA pinpoints the requirements engineers to the portions of the legal document where they need to invest their efforts and time.

We illustrate in Figure 1, the QA assistance provided by COREQQA to a requirements engineer, who is interested in understanding the regulations related to personal data breach. The example question is specifically related to the process for handling personal data breaches. The answer to this question is mined in the GDPR text. The legal obligations with regard to handling data breaches can have a significant impact on the software development process, e.g., sending notifications to different responsible agents within legally-binding time constraints. COREQQA assists the requirements engineer in retrieving relevant information to define the

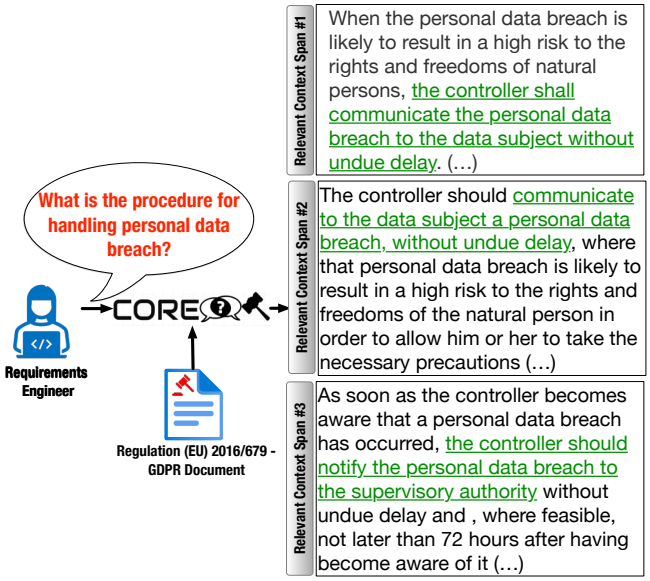

Figure 1: Example of COREQQA's QA assistance on GDPR [11].

respective compliance requirements for handling data breaches for the software system under development. As we elaborate in Section 2, COREQQA returns as output the top-N[1] relevant context spans from a given legal document and the potential answers highlighted in the context spans. COREQQA builds on large-scale natural language processing (NLP) language models for solving the QA task (also widely known as *machine reading comprehension (MRC) task* [14]). QA models for MRC generally assume that for each question, the relevant context span (containing the answer) is known a priori. Developing a practical QA tool with this restriction is infeasible, as requirements engineers have no means of knowing the exact context span with the correct answer in advance. Therefore, in COREQAA we first find top-N relevant context spans which likely contain the answer. To do so, we compute the semantic similarity between each context span in the legal document and the input question. Then, we demarcate the answer to the input question using the QA models.

We further observe that information relevant to answering the question could be found in multiple non-contiguous context spans (i.e., different sections in the same legal document). For example, the top-3 context spans selected in Figure 1 are all directly relevant for answering the question. The first two spans (retrieved from different sections of the document) explain the process of communicating breach details to the data subject, while the third span specifies how to communicate breach details to the supervisory authorities. Thus, by retrieving multiple relevant spans and further highlighting the likely answers, COREQQA enables the requirements engineer to specify a complete and precise set of compliance requirements. We leave configuring the N parameter to the requirements engineer. While selecting higher values of N entails more time and effort for reviewing the retrieved context spans, we believe that using COREQQA is still much more cost-effective in practice than manually traversing the entire legal document for the answer.

In the remainder of this tool demonstration paper, we elaborate the architecture of the tool, the dataset that we generated for developing COREQQA as well as an end-to-end usage scenario.

2 TOOL ARCHITECTURE

The end-to-end architecture of COREQQA is illustrated in Figure 2. COREQQA aims at answering a given question posed by a requirements engineer in NL on some legal document. Below, we elaborate the main steps of the tool marked as 1 – 3 in Figure 2. We implemented COREQQA in Python 3.8.

2.1 Text Preprocessing

In the first step, COREQQA parses the legal document and applies a simple NLP pipeline which is composed of tokenization and sentence splitting. The tool then applies a set of regular expressions for normalizing the text (e.g., removing periods from the ending of acronyms, "Art." becomes "Art"). The motivation for normalizing the text is to improve the accuracy of sentence splitting. We operationalize the NLP pipeline using NLTK library [6, 17], and the *re module* in Python for regular expressions[2]. In this step, we further partition the legal document into context spans. Due to technical constraints of underlying QA models, the maximum size of each context span is 512 tokens. To maintain coherence, we split the document into paragraphs first, and then check their size. Each paragraph fitting this size limitation is regarded as one context span. Otherwise, we split the paragraph into half, and check the size again. This process is iteratively performed until size limitations are met. The output of this step is the list of context spans representing the input legal document.

2.2 Relevant Context Retrieval

In the second step, COREQQA computes the semantic similarity between the input question and each context span generated from the previous step. We implement this step using the BERT cross-encoder (BCE) model available in the Sentence-Transformer 2.1.0 [19] provided by Hugging Face[3]. BCE takes as input two text fragments, and returns as output a score between 0 and 1 indicating how semantically similar the two fragments are, with 1 being identical. To assess the relevance of the context span, we first compute BCE between the question and each sentence in the context span and then assign to the context span the maximum score achieved by any sentence. The intuition behind this computation strategy is that only a portion of the context span is expected to contain the likely answer to the input question.

Once we compute a score per context span, we rank the spans in descending order. We do this using the sort function from *pandas* in Python[4]. The result of this step is a list of top-N relevant context spans to the input question. We keep the value N as a parameter that can be initialized by the requirements engineer. The value of N depends on the practical context in which COREQQA is applied.

[1] N is a configurable parameter and is set $N = 3$ for the example question in the figure

[2] https://docs/python.org/3.8/library/re/

[3] https://huggingface.co/

[4] https://pandas.pydata.org

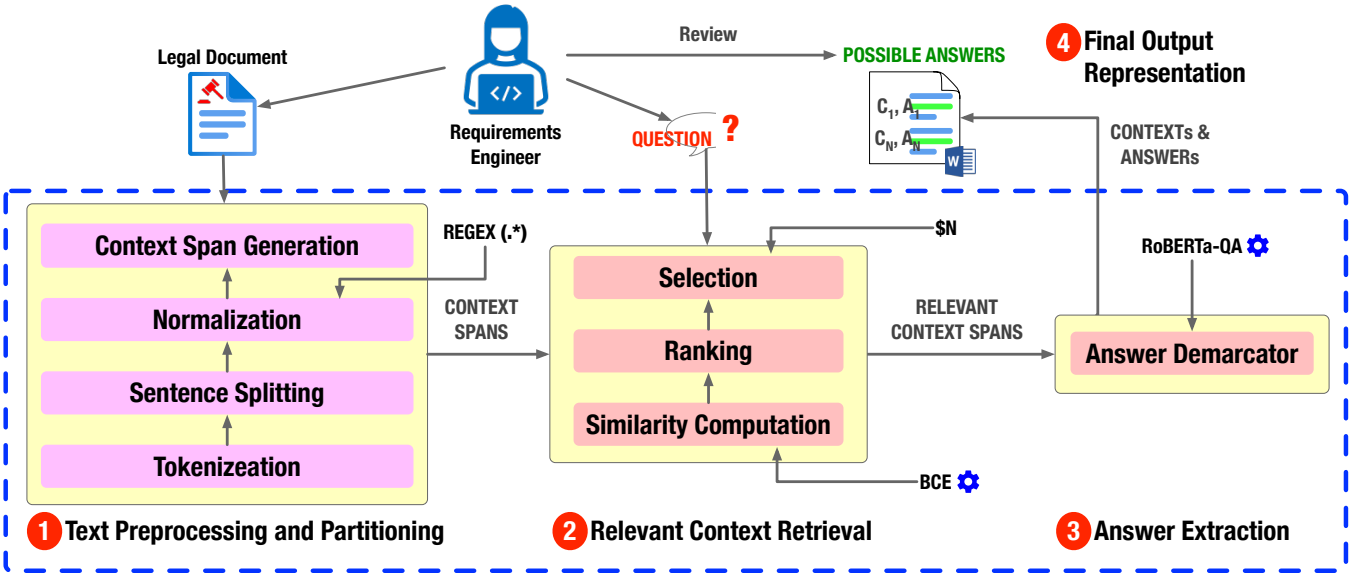

Figure 2: Overview of the end-to-end architecture of COREQQA.

Selecting large values (e.g., ≥ 10) entails that the requirements engineer will review many context spans to gain a better understanding of the compliance requirements associated with the question. Selecting small N values (e.g., ≤ 3) entails less context spans to be reviewed, but also a higher risk for the right answer not to be found in any of the top-N relevant spans. In Figure 1, we show an example of top-3 context spans retrieved in this step. The default value of the configurable parameter N is set to 5 in COREQQA, based on previous experiments [2].

2.3 Answer Extraction

In the last step, we pass on the top-N context spans deemed relevant in the previous step together with the input question to a pre-trained QA model. In our work, we apply the Transformers library to extract answers for the given question using the RoBERTa QA model (*roberta-base-squad2-distilled*). RoBERTa then extracts from each context span a potential answer for the input question. In Figure 1, we highlight the extracted answers in green. We note that the engineer has access to the context spans already in the previous step. Thus, this step is not essential for providing assistance to the requirements engineer in understanding the questions related to compliance requirements. However, highlighting the answer in the context span improves readability and leads to a more efficient reviewing process. In practice, when the engineer selects a larger number N (say 10), it is then advantageous to demarcate the answer automatically to help the engineer quickly navigate through the context spans.

2.4 Final Output Representation

For presenting the final output of the tool to the requirements engineer, we export for each question the top-N context spans and the highlighted answers within these context spans as a Microsoft Word document. The document also shows for each context span a confidence score (range 0–1) of the answer highlighted in the span. This confidence score is automatically assigned to the extracted answer by the QA model. We use the python-docx library v0.8.11 (https://python-docx.readthedocs.io) for exporting the output and visualizing highlighted answers in the relevant top-N context spans.

3 EVALUATION

COREQQA has been evaluated on four legal documents, wherein the question-answer pairs were identified by two experts – one expert in legal informatics and the other in requirements engineering [2]. In the following, we describe the four documents:

- ***GDPR*** or General Data Protection Regulation (EU) 2016/679 is the European privacy law that harmonises the data protection, privacy and personal data transfer requirements [11]. The experts identified 36 question-answer pairs from the entire document. The document was partitioned in 301 context spans by the "Text Preprocessing" step of Figure 2.
- ***Directive (EU) 2019/770*** is the European directive for regulating the supply of digital content or digital services, and laying down rules for contracts between any trader and consumer of digital content or service [9]. The experts identified 33 question-answer pairs in this document, and the document was split into 120 context spans.
- ***Directive (EU) 2019/771*** is the European directive concerning the sale of goods [10]. Directive (EU) 2019/771 complements Directive (EU) 2019/770, as it formalises the contracts on the sale of goods that contain digital elements that require digital content or service. For example, the regulations related to the contracts of the smartphone are covered by Directive (EU) 2019/771, whereas the regulations for operating systems or apps on the smartphone might be covered by Directive (EU) 2019/770. The experts identified 19 question-answer

pairs in Directive (EU) 2019/771, and the document text was split into 102 context spans.

• ***Luxembourg Law of 25 March 2020*** is an amendment to numerous existing finance and banking related laws in Luxembourg. The amendment was intended to set due diligence measures for a central electronic data retrieval system related to bank accounts and safe-deposit boxes in Luxembourg, and was a step towards tackling money laundering [1]. The experts identified 19 question-answer pairs in this legal document. The document text was split into 23 context spans.

To identify the most accurate similarity metric for context span retrieval, we compared BCE (Section 2.2) with TF-IDF similarity [3] – a metric commonly used in the NLP domain. BCE was significantly more accurate than TF-IDF in our experiments. Overall, from the 107 questions over the four documents, BCE retrieves the correct context span for 100 questions (for the top-5 spans). In addition to RoBERTa, we further evaluated three QA models, namely BERT [8], ALBERT [15], ELECTRA [7] for answer extraction (Section 2.3). RoBERTa was deemed the most accurate as it correctly extracted the answers for 97 questions.

We also analyzed the questions where COREQQA did not highly rank the correct context span (within top-5) or RoBERTa model did not extract the correct answer. Our analysis showed that generic questions, such as the ones formulated for defining or elaborating on a legal concept, were not correctly answered. This is because the legal document (or even a given context span) would usually contain several instances of such legal concept, thus misleading both the context span retrieval and answer extraction steps of COREQQA. We also realized that complicated questions (e.g., with composite conditions) were difficult to answer for COREQQA.

Last but not least, COREQQA answers questions within reasonable execution time. Thus, in short, our evaluation indicates that COREQQA produces accurate results and is fit for use by requirements engineers in practice. For answering one question from a legal document including an average of 620 sentences, COREQQA requires a total of ≈34 seconds.

4 USAGE SCENARIO

In this section, we describe an end-to-end example illustrating how our tool can be applied in practice by a requirements engineer. Let *KoopaApp* be a new system under development. KoopaApp is a gym fitness app for maintaining users' workout information and other health-related data. KoopaApp accesses personal information such as the location from other apps on the users' smartphone. During the pandemic, such applications often raised concerns about privacy. For example, several health applications were analyzed for privacy-related issues in the RE literature [4, 12].

A requirements engineer (*Daisy*) is in charge of specifying the KoopaApp requirements, including compliance requirements. As an example, we focus only on a subset of compliance requirements related to privacy and data protection. Daisy (as is often the case in most software projects) is not very familiar with the privacy regulations, yet she knows well the functionalities and characteristics of the KoopaApp. During the elicitation of requirements, Daisy identifies a set of functionalities that make use of personal data and are thus subject to compliance. Some of these functionalities are related to the security of collected personal data. We assume here that Daisy or her team are aware of the relevant legal documents for their project. Suggesting relevant documents is beyond the scope of COREQQA. Accounting to possible security threats, Daisy poses a question ("What is the procedure for handling a personal data breach?") using the COREQQA tool on the GDPR [11]. COREQQA in turn provides the output shown in Figure 1.

From the output of COREQQA, Daisy is able to formulate the following compliance requirements (prefixed with the ID *CR*) under the label *Users Data Breach*.

Notify Users about the Data Breach.

CR_1. If a data breach is identified on the KoopaApp server, the KoopaApp-NotifyService shall inform the affected users.

CR_2. The KoopaApp-NotifyService shall email the affected users on the registered email address and store the 'user informed' response on the server.

CR_3. The KoopaApp-NotifyService shall notify the affected users on the app and store the 'read' response on the server.

Notify the CIO about the Data Breach.

CR_4. If a data breach is identified on the KoopaApp server, the KoopaApp-NotifyService shall send an email to the Chief Information Officer notifying the breach, within 72 hours of its occurrence.

The four compliance requirements fulfill the regulations provided in Figure 1.

5 CONCLUSION

We presented COREQQA – a tool for assisting requirements engineers in better understanding compliance requirements through question-answering based on regulatory or legal documents. COREQQA is developed using a manually created dataset that combines a joint effort of a requirements engineer and a legal expert over four diverse legal documents. The tool is based on recent large-scale language models that are pre-trained for question-answering. Specifically, the tool applies the Sentence BERT cross encoder for retrieving the most relevant text passages from a legal document for a given question. The tool further employs the RoBERTa question-answering model for highlighting the likely answers to the question in the retrieved text passages.

In future, we plan to conduct a user study to assess how useful COREQQA is in practice.

ACKNOWLEDGMENTS

This paper was supported by Central Legislative Service (SCL) – Government of Luxembourg, the Luxembourg National Research Fund (FNR) under grants PUBLIC2-17/IS/11801776, and by NSERC of Canada under the Discovery and CRC programs.

REFERENCES

[1] 2020. Law of 25 March 2020 (coordinated version) establishing a central electronic data retrieval system related to IBAN accounts and safe-deposit boxes. https://www.cssf.lu/en/Document/law-of-25-march-2020-data-retrieval/

[2] Sallam Abualhaija, Chetan Arora, Amin Sleimi, and Lionel Briand. 2022. Automated Question Answering for Improved Understanding of Compliance Requirements: A Multi-Document Study. In *30th IEEE International Requirements Engineering Conference (RE'22)*.

[3] Akiko Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing & Management* 39, 1 (2003), 45–65. https://doi.org/10.1016/S0306-4573(02)00021-3

[4] Muneera Bano, Chetan Arora, Didar Zowghi, and Alessio Ferrari. 2021. The Rise and Fall of COVID-19 Contact-Tracing Apps: when NFRs Collide with Pandemic. In *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE, 106–116. https://doi.org/10.1109/RE51729.2021.00017

[5] Brian Berenbach, Daniel J Paulish, Juergen Kazmeier, and Arnold Rudorfer. 2009. *Software & systems requirements engineering: in practice*. McGraw-Hill Education.

[6] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly.

[7] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *CoRR* abs/2003.10555 (2020). https://doi.org/10.48550/arXiv.2003.10555 arXiv:2003.10555

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). https://doi.org/10.48550/arXiv.1810.04805 arXiv:1810.04805

[9] EU (2019/770). 2019. Directive (EU) 2019/770 of the European Parliament and of the Council of 20 May 2019 on certain aspects concerning contracts for the supply of digital content and digital services, OJ L 136, 22.5.2019, p. 1–27. http://data.europa.eu/eli/dir/2019/770/oj

[10] EU (2019/771). 2019. Directive (EU) 2019/771 of the European Parliament and of the Council of 20 May 2019 on certain aspects concerning contracts for the sale of goods, amending Regulation (EU) 2017/2394 and Directive 2009/22/EC, and repealing Directive 1999/44/EC, OJ L 136, 22.5.2019, p. 28–50. http://data.europa.eu/eli/dir/2019/771/oj

[11] EU (GDPR). 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ L 119, 4.5.2016, p. 1–88. http://data.europa.eu/eli/reg/2016/679/oj

[12] Mattia Fazzini, Hourieh Khalajzadeh, Omar Haggag, Zhaoqing Li, Humphrey Obie, Chetan Arora, Waqar Hussain, and John Grundy. 2022. Characterizing Human Aspects in Reviews of COVID-19 Apps. In *9th IEEE/ACM International Conference on Mobile Software Engineering and Systems*. https://doi.org/10.1145/3524613.3527814

[13] Marijn Janssen, Paul Brous, Elsa Estevez, Luis S Barbosa, and Tomasz Janowski. 2020. Data governance: Organizing data for trustworthy Artificial Intelligence. *Government Information Quarterly* 37, 3 (2020), 101493. https://doi.org/10.1016/j.giq.2020.101493

[14] Dan Jurafsky and James H. Martin. 2020. *Speech and Language Processing* (3rd ed.). https://web.stanford.edu/~jurafsky/slp3/ (visited on 2022-01-04).

[15] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *CoRR* abs/1909.11942 (2019). https://doi.org/10.48550/arXiv.1909.11942 arXiv:1909.11942

[16] Dorothy E Leidner and Olgerta Tona. 2021. The CARE Theory of Dignity Amid Personal Data Digitalization. *MIS Quarterly* 45, 1 (2021).

[17] Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*.

[18] Paul N Otto and Annie I Antón. 2007. Addressing legal requirements in requirements engineering. In *15th IEEE international requirements engineering conference (RE 2007)*. IEEE, 5–14. https://doi.org/10.1109/RE.2007.65

[19] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *CoRR* abs/1908.10084 (2019). https://doi.org/10.48550/arXiv.1908.10084 arXiv:1908.10084

[20] Amin Sleimi, Marcello Ceci, Nicolas Sannier, Mehrdad Sabetzadeh, Lionel Briand, and John Dann. 2019. A Query System for Extracting Requirements-Related Information from Legal Texts. In *27th IEEE International Requirements Engineering Conference*. IEEE. https://doi.org/10.1109/RE.2019.00041

[21] Amin Sleimi, Nicolas Sannier, Mehrdad Sabetzadeh, Lionel Briand, and John Dann. 2018. Automated Extraction of Semantic Legal Metadata using Natural Language Processing. In *Proceedings of the 26th IEEE International Requirements Engineering Conference*. https://doi.org/10.1109/re.2018.00022