Automated Question Answering for Improved Understanding of Compliance Requirements: A Multi-Document Study

Sallam Abualhaija[*§], Chetan Arora[‡*], Amin Sleimi[*], Lionel C. Briand[*†]
[*]SnT Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg
[‡]Deakin University, Geelong, Australia
[†]School of Electrical Engineering and Computer Science, University of Ottawa, Canada
Email: sallam.abualhaija@uni.lu, chetan.arora@deakin.edu.au, amin.sleimi@uni.lu, lbriand@uottawa.ca

***Abstract*—Software systems are increasingly subject to regulatory compliance. Extracting compliance requirements from regulations is challenging. Ideally, locating compliance-related information in a regulation requires a joint effort from requirements engineers and legal experts, whose availability is limited. However, regulations are typically long documents spanning hundreds of pages, containing legal jargon, applying complicated natural language structures, and including cross-references, thus making their analysis effort-intensive. In this paper, we propose an automated question-answering (QA) approach that assists requirements engineers in finding the legal text passages relevant to compliance requirements. Our approach utilizes large-scale language models fine-tuned for QA, including BERT and three variants. We evaluate our approach on 107 question-answer pairs, manually curated by subject-matter experts, for four different European regulatory documents. Among these documents is the general data protection regulation (GDPR) – a major source for privacy-related requirements. Our empirical results show that, in ≈94% of the cases, our approach finds the text passage containing the answer to a given question among the top five passages that our approach marks as most relevant. Further, our approach successfully demarcates, in the selected passage, the right answer with an average accuracy of ≈91%.**

***Index Terms*—Requirements Engineering, Regulatory Compliance, Natural Language Processing (NLP), Question Answering, Language Models (LMs), BERT.**

I. Introduction

Requirements engineering (RE) is an essential process in software development, notably for specifying, verifying and maintaining the functional and non-functional requirements of a system-to-be [1]. In this digital age, systems heavily rely on personal data and other types of confidential information [2]. Consequently, new regulations are enforced to ensure the protection of personal data, provide guarantees to consumers, and meet certification requirements. Software systems as well as organizations must comply with these regulations lest they confront serious consequences, such as, product recall, sanctions, and loss of business and reputation. For instance, the European Commission requires all parties involved in the supply of digital content and services to comply with Directive (EU) 2019/770 [3].

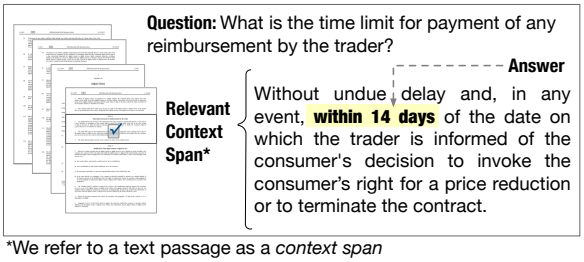

Fig. 1. Example of QA Assistance in Directive (EU) 2019/770.

As a result, nowadays, requirements engineers have to deal with compliance requirements, i.e., ensure the system's compliance with relevant regulations. Extracting compliance requirements from regulations is an important but challenging task for multiple reasons. First, this process is time consuming and error-prone since regulations often contain complicated natural language (NL) structures, are composed of long chapters, and contain many references to external articles. Second, requirements engineers do not necessarily have sufficient legal expertise to efficiently navigate through the regulations and find information related to compliance requirements. In addition, relying solely on legal experts is impractical, since they cannot identify what is relevant for the compliance of the system without inputs from requirements engineers. Ensuring the availability of both legal experts and requirements engineers for joint manual elicitation sessions is however difficult, and consulting legal experts can be expensive.

Question-answering (QA) can be used as a means to facilitate compliance-related information extraction from regulations. For illustration, consider the simplified example in Fig. 1. Let *Nord Ltd* (also called *the trader*) be an organization that offers subscriptions to music streaming through an online platform *norskstreaming.no*. The requirements engineers at *Nord Ltd* have to account for compliance requirements in Directive (EU) 2019/770. For in-

stance, the subscription system shall allow a consumer to unsubscribe at any time, i.e., terminate the digital service contract. Following this, *Nord Ltd* is obliged to enable (pro-rata) reimbursement to the consumer within 14 days of the cancellation. This regulatory constraint can be retrieved from the Directive using QA as shown in Fig. 1. To protect the trader's interests, the requirements engineer can use the information extracted for writing a compliance requirement. For example, this requirement can be formulated in NL as: *"When a consumer contract is terminated, the norskstreaming payment module shall trigger a reimbursement to the consumer within a maximum time of 14 days pursuant to Article 18 in Directive (EU) 2019/770"*.

The use of QA for specifying compliance requirements has not been sufficiently investigated in RE. The only directly related work is the one by Sleimi et al. [4] who propose a query system using semantic web technologies over taxation laws. The authors built on previous work [5] wherein they extract semantic metadata (e.g., actor, time, and sanction) from regulatory documents. The answer to a query must constitute some of these semantic metadata. Sleimi et al.'s work has two limitations. First, intensive manual work is required for generating the semantic web-related schema and queries. Second, their work does not handle any query whose answer cannot be mapped to the metadata. Existing work on QA for the legal domain in the NLP literature [6], [7], [8], [9] mostly focuses on improving the understandability of legal texts for users. Existing NLP approaches do not consider the needs of requirements engineers with respect to compliance requirements extraction.

As a step toward addressing the limitations outlined above, we propose an automated QA approach to assist requirements engineers in finding compliance-related information in regulations. Our approach aims at (1) providing a surrogate in the absence of a legal expert, and (2) drastically reducing time and effort through automated assistance. Our approach builds on recent advances in natural language processing (NLP) technologies. QA in NLP is the task of automatically finding the most likely answer to a question posed in NL in a given text passage. In our work, we refer to a single text passage as a *context span*. In our study, QA entails two sub-tasks: (1) *information retrieval-based (or IR-based)* which is concerned with retrieving the context span(s) from a collection of documents where the answer to an input question is highly likely to be found, and (2) *machine reading comprehension (or MRC)* that involves highlighting the answer in the retrieved span [10]. The recent QA solutions are based on large-scale language models (e.g., BERT – introduced in Section II) and focus exclusively on MRC [11]. Such models are trained with the assumption that the relevant context span containing the right answer is known a priori for each question. This assumption is not practical in our case, since a requirements engineer has no means of knowing in advance the location of the right answer to a question in a regulatory document. To provide practical QA assistance to requirements engineers and further leverage the recent NLP technologies for QA, we propose a novel approach that combines the two above-mentioned sub-tasks.

Our approach extracts compliance-related information, given a regulatory document and a question posed in NL, in two phases. Phase I employs IR-based methods to retrieve the top-k context spans which are relevant to the input question. Phase II solves MRC, i.e., highlights the likely answers to the question in the retrieved context spans. The output of the approach is then presented to the requirements engineer. For simplicity, we show in the example in Fig. 1 the output of our approach including the top-ranked context span ($k = 1$) and the answer highlighted in that context span.

Contributions. This paper makes the following contributions:

(1) We propose QA automation that (a) draws on recent NLP technologies, and (b) assists requirements engineers in extracting information from regulatory documents. The requirements engineers can simply pose questions in NL about matters that are relevant to compliance requirements and our approach finds the answers in the regulatory document. We elaborate our approach in Section III;

(2) We empirically evaluate our approach on a QA dataset curated by two experts from the legal and RE domains. The dataset covers four European regulatory documents which are relevant to software systems, including Directive EU 2019/770 and the general data protection regulation (GDPR) – a major source for privacy-related requirements. Our QA dataset is composed of 107 questions and their respective answers as we further explain in Section IV-C. We make this dataset available to the RE community as online annex [12]. Over this dataset, our approach has an average recall of 93.5% in finding the context span that contains the right answer among the top five spans marked as relevant to the input question. Our approach further identifies the answer to the input question in the retrieved context span with an average accuracy of 90.7%.

Structure. Section II provides the background of our automated solution. Section III describes our QA approach. Section IV evaluates our approach. Section V discusses threats to validity. Section VI reviews the related work and Section VII concludes the paper.

II. Background

In this section, we introduce the background related to our QA approach, including language models and text similarity.

Language Models (LMs). Language Modeling is an NLP task that focuses on determining the probability distribution of word sequences in NL [10]. Given a sequence of words, an LM predicts the most likely next word [13]. For instance, an LM would predict "wine" as the most likely next word in the input sequence "He was drinking red [WORD]". To accurately estimate the probability distributions of words, an LM is trained on large corpora of texts. Below, we discuss BERT – one of the recent and widely-used LMs in the NLP community, and three variants that we explore in our work.

(1) *Bidirectional Encoder Representations from Transformers (BERT)* [14] is a transformer-based model that learns the sequential structure of the text in both directions, i.e., left to right and right to left. A transformer is a multi-layer encoder-decoder architecture, wherein the encoder converts the input text into an intermediate numerical vector representation, and the decoder converts the vector into output text. BERT is an encoder that is (pre-)trained using two objectives, namely masked language modeling (MLM) and next sentence prediction (NSP). MLM randomly masks a fraction of tokens in the training corpus, i.e., BooksCorpus [15] and English Wikipedia. In contrast to statistical LMs which predict the next word, this objective enables the prediction of a missing word in a sequence of words. For example, BERT predicts "wine" as the masked word in the input sequence "He was drinking red [MASK] and eating cheese.". The NSP objective is concerned with predicting whether two sentences are consecutive or not. The original BERT model has 12 layers of encoders with 110 million parameters and a vocabulary of 30,000 words.

(2) *A Lite BERT (ALBERT)* is a light-weight variant of BERT [16]. To improve efficiency and lower the computational resources required by BERT, ALBERT applies two strategies to reduce the number of parameters in the original BERT model. The first strategy is to enable the sharing of parameters across different encoder layers. The second strategy is to decompose larger matrices representing the parameter space into smaller ones. ALBERT has a total of 12 million parameters with the same vocabulary as BERT.

(3) *A Robustly Optimized BERT (RoBERTa)* [17] is another variant of BERT, which expands the text body used for pre-training BERT with additional datasets, namely CC-NEWS [18], OpenWebText [19] and Stories dataset [20]. In total, RoBERTa is trained on 161GB of text, compared to the 16GB used for pre-training BERT. Other factors that set RoBERTa apart from BERT are: (i) longer training time with bigger batches; (ii) training on longer text sequences; and (iii) training MLM using dynamically-changing masking patterns. RoBERTa has 125 million parameters and a vocabulary of 50,000 words.

(4) *Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA)* is a variant of BERT which is, compared to BERT, pre-trained with the objective of replaced token detection (RTD) instead of MLM [21]. RTD replaces a masked token with another token instead of masking it. For instance, the token "the" and "cooked" are replaced with "a" and "ate" in the sentence "The chef cooked the meal". ELECTRA is then trained to predict whether each token in the input sequence is original or replaced. ELECTRA has the same number of parameters and vocabulary size as BERT.

LMs for QA. A large-scale pre-trained LM can be fine-tuned to solve different downstream NLP tasks, e.g., QA and computing text similarity [22]. The models we employ in this work based on BERT and its variants are fine-tuned to address machine reading comprehension (MRC). MRC is a challenging task which involves reading a text passage and then answering questions about it [10], [23]. The fine-tuning of BERT and its variant models is performed by exposing the pre-trained models to a large dataset annotated for MRC. A common example of such a dataset is SQuAD (stanford question answering dataset) [24] which is composed of 100K question-answer pairs manually collected from Wikipedia articles through crowdsourcing.

Recall that BERT has been pre-trained to address the NSP objective, i.e., predicting whether two sentences A and B are consecutive. Fine-tuning retains this objective but replaces sentences A and B with the input question and the text passage. The fine-tuned model then learns to identify, in the selected text passage, for each input question, the first and last tokens of the predicted answer. As we elaborate in Section III, we apply the fine-tuned models of the above-listed LMs (referred to as *QA models*). These models have a constraint on the size of the input text passage to be less than or equal to 512 tokens. When a question cannot be answered from the text passage, QA models often randomly guess and identify answer in the passage. Due to the complexity of MRC, training machines to recognize unanswerable questions is still an ongoing research task.

Text Similarity. The QA models explained earlier assume that the text passage containing the answer to a question is known in advance. Thus, an essential prior step is to find the text passage that likely contains the answer. This is often done using IR-based methods which select a relevant text passage from a collection of passages according to their similarity with the input question. In our work, we apply two

widely-used text similarity methods explained below. (1) *Cosine similarity over term frequency-inverse document frequency (TF-IDF) vectors:* TF-IDF is a method used to represent text (often a document in a collection of documents) with numeric vectors [25]. The TF-IDF score of a term is computed as the frequency of that term in the document (TF) normalized by the inverse document frequency (IDF), which represents how common the term is across all documents in the collection. TF-IDF assigns a higher score to a term if it is "important" to that document, i.e., it occurs often, but rarely appears in the other documents in the collection. The TF-IDF vector representation can be used to measure the similarity of two text sequences. In our work, we compute the cosine similarity which is the dot product of the TF-IDF vectors [26]. Thereafter, we refer to this metric as *CS-TFIDF*, which takes a value between 0 and 1, where 0 and 1 indicate entirely dissimilar and identical sequences, respectively [26]. (2) *BERT cross-encoder-based similarity (BCE):* BCE measures the similarity of two sentences based on Sentence-BERT (SBERT), a variant of BERT optimized for representing sentences instead of tokens [22]. BCE takes as input the textual content of two sentences. Then, BCE returns a value between 0 and 1 based on the similarity between the sentences' representations as produced by SBERT. This value, similar to CS-TFIDF, indicates how similar the sentences are, where 1 indicates identical sentences.

III. Approach

Fig. 2 provides an overview of our QA approach which is composed of three steps. The input to the approach is a *regulatory document* ($RegD$) and a question ($\mathbf{q}$) posed in NL. In step A, we preprocess $RegD$ and $\mathbf{q}$ using an NLP pipeline. In this step, we further partition $RegD$ into a set of context spans. In step B, we rank the context spans according to their relevance to $\mathbf{q}$. We then select the top-k relevant spans which are likely to contain the answer. In step C, we apply a QA model to extract a likely answer to $\mathbf{q}$ from each relevant context span. Our approach returns as output the relevant context spans from step B together with the likely answers extracted in step C. Below, we elaborate on these steps.

A. Preprocessing

For preprocessing $RegD$ and $\mathbf{q}$, we use a simple NLP pipeline of two modules, namely a *tokenization* module to split the text into tokens such as words and punctuation marks, and a *sentence splitting* module to delineate the sentences in the input text.

In this step, we further partition $RegD$ into a list of context spans, denoted as $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$, where each span $c_i \in \mathcal{C}$ has a maximum length of 512 tokens. Recall from Section II that this limit is imposed by the QA models employed in step C. These models are fine-tuned to address MRC, where, by definition, the answer is extracted from a single text passage (i.e., a context span). Irrespective of this size limit, partitioning $RegD$ into context spans is essential in our approach, since some context spans will be provided as a part of the output. In any case, reasonably sized context spans will render them easier to review by a requirements engineer.

We automatically generate $\mathcal{C}$ in two steps. The first step applies regular expressions over the annotations produced by the NLP pipeline in order to identify the articles in $RegD$ based on its formatting structure. A regulatory document can be structured in multiple chapters which consist of multiple sections organized in articles. We are mainly interested in splitting $RegD$ into articles, since each article often discusses a coherent topic in a regulation. For example, under Section 2 "Security of Personal Data" in GDPR, Article 32 stipulates the regulations concerning the *security of processing* whereas Article 33 is about *notification of a personal data breach to the supervisory authority*. In the second step, we check the size of each article identified in $RegD$. If the number of tokens in the article are less than or equal to 512, then the entire article is regarded as a context span and added to $\mathcal{C}$. Otherwise, a longer article is recursively split until a coherent text sequence (i.e., a paragraph or sentence) can be achieved within the size limit. The generated set $\mathcal{C}$ and $\mathbf{q}$ are passed on to step B.

B. Relevant Context Span Selection

In this step, we rank the context spans in $\mathcal{C}$ according to their *relevance* to $\mathbf{q}$. A context span $c_i \in \mathcal{C}$ is deemed relevant to $\mathbf{q}$ if c_i and $\mathbf{q}$ are semantically similar. Larger similarity values indicate that c_i is of a higher relevance to $\mathbf{q}$, and thus may contain the likely answer. As we show in Section IV, we experiment with two alternative metrics for computing text similarity (discussed in Section II), namely CS-TFIDF and BCE.

For computing CS-TFIDF, we first generate two TF-IDF vectors; one representing $\mathbf{q}$ and the other representing a context span $c_i \in \mathcal{C}$. We define the vocabulary set as the union of all terms occurring in both $\mathbf{q}$ and $\mathcal{C}$. The vocabulary size determines the dimension of the resulting TF-IDF vectors. In this setting, the context spans $c_i \in \mathcal{C}$ are regarded as distinct documents. We compute the TF-IDF value for each term in the vocabulary set. The vector representing $\mathbf{q}$ consists of the TF-IDF value of each term occurring in $\mathbf{q}$ and zeros for all other terms in the vocabulary set. Similarly, the vector representing c_i contains the TF-IDF values of the terms occurring in c_i. Finally, we assess relevance between $\mathbf{q}$ and c_i by measuring the cosine similarity between the respective TF-IDF vectors.

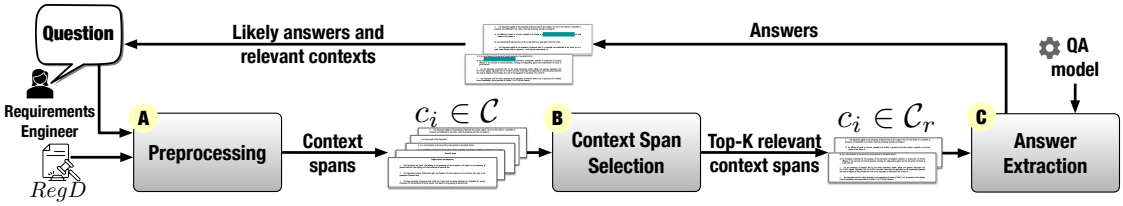

Fig. 2. Approach Overview.

Unlike CS-TFIDF, BCE measures the similarity directly on the textual content of two sentences without having to generate vectors. Since c_i can contain multiple sentences, we compute a BCE score for each sentence in c_i against $\mathbf{q}$. We then take the maximum value as the final score between $\mathbf{q}$ and c_i. The intuition for taking the maximum (instead of average) is that only few sentences in c_i may be relevant to $\mathbf{q}$, when c_i contains the likely answer. Averaging the BCE scores over all sentences in c_i will often yield low similarity and thus mislead the ranking of c_i.

The result of applying a text similarity metric is a score assigned to each $c_i \in \mathcal{C}$ indicating how relevant c_i is to $\mathbf{q}$. We simply sort these scores in descending order and select the top-k *relevant context spans*, denoted as $\mathcal{C}_r \subset \mathcal{C}$. We discuss the implications of setting the k value in Section IV.

C. Answer Extraction

In the last step, we iteratively feed $\mathbf{q}$ and each $c_i \in \mathcal{C}_r$ to a QA model to extract a likely answer to $\mathbf{q}$ from c_i. The final output of our approach consists of, for each posed $\mathbf{q}$, a set of relevant context spans $\mathcal{C}_r$, where in each $c_i \in \mathcal{C}_r$ a likely answer to $\mathbf{q}$ is highlighted. As we discuss in Section IV, we experiment with four alternative QA models, namely BERT, ALBERT, RoBERTa and ELECTRA (introduced in Section II). The application of the QA model in this step has two constraints. First, the extracted answer is always a short text span in c_i (e.g., a part of a sentence). Second, the QA model always finds an answer to $\mathbf{q}$ in any given c_i, whether it is the right context span or not. The reason in both cases is that the fine-tuned models for QA in NLP are optimized to address MRC, that is extracting a short answer to some question from a given text passage. Despite such constraints, we believe that highlighting a short answer in each $c_i \in \mathcal{C}_r$ is beneficial for a requirements engineer and provides them with guidance when looking for compliance-relevant information in the regulation.

IV. Evaluation

This section discusses the empirical evaluation of our approach.

A. Research Questions (RQs)

RQ1. Which text similarity metric is the most accurate for selecting context spans relevant to a given question? As discussed in Step B of our approach (Section III-B), we experiment with two metrics – CS-TFIDF and BCE – for ranking the context spans according to their relevance to the input question. RQ1 aims to determine the metric that yields the most accurate results for selecting the top-k relevant context spans that will be used for answering the input question. The most accurate metric will be used to answer the subsequent RQs.

RQ2. Which QA model yields the most accurate answer extraction results? As discussed in Step C (Section III-C), we experiment with four alternative QA models, namely BERT, ALBERT, RoBERTa and ELECTRA. RQ2 investigates the accuracy of these models in extracting the correct answer from the top-k context spans selected in Step B of our approach.

RQ3. What is the execution time of our approach? From a practical standpoint, our approach shall assist requirements engineers by finding compliance-related information from a given regulatory document in a practical time. RQ3 reports the execution time of each step of our approach, and investigates whether our approach is practical in terms of running time.

B. Implementation and Availability

Our approach (steps A-C in Fig. 2) is implemented in Python 3.8. The preprocessing in Step A is implemented using the Natural Language Toolkit (NLTK) library 3.5 [27] and Python's regular expression (*re*) module (https://docs.python.org/3.8/library/re). We use Scikit-learn 0.23.2 [28] for calculating CS-TFIDF in the context span selection (step B). We calculate BCE using the sentence-transformers library 2.1.0 [22] based on the Transformers 4.6.1 library [29] provided by Hugging Face (https://huggingface.co/). To rank the context spans in step B, we apply the *heapq* module in Python (https://docs.python.org/3.8/library/heapq.html). In step C, we use the Transformers library to extract answers for given questions using QA models. Specifically, the QA models which we employ in our experiments include *bert-base-cased-squad2* for BERT, *albert-base-v2-squad2* for ALBERT, *roberta-base-squad2-distilled* for RoBERTa, and *electra-base-squad2* for ELECTRA. All these models are available in the Transformers library. We implement our approach using Jupyter Notebooks [30] and make it publicly available as online annex [12].

C. Data Collection Procedure

The goal of our data collection is to manually propose questions and extract their respective answers from regulatory documents. To do that, we had two subject-matter experts review four regulatory documents (explained next). The first expert (Mario – pseudonym) has a PhD in legal informatics and more than eight years of experience in the legal domain. The second expert (Luigi – pseudonym), who is a co-author, has more than five years experience as a requirements engineer in several industrial sectors. Through his industrial career, Luigi was responsible for eliciting and specifying compliance requirements.

We collected our data from four software systems-relevant European regulatory documents, referred to as $RegD_1$ – $RegD_4$ and described below.

- $RegD_1$ is GDPR (Regulation (EU) 2016/679) – the European regulation on data protection, privacy and personal data transfer [31].
- $RegD_2$ is the European directive for regulations on the supply of digital content and digital services (Directive 2019/770) [3].
- $RegD_3$ is the European directive on the sales of goods (Directive 2019/771) [32]. $RegD_3$ complements $RegD_2$ as the former specifies regulations on contracts for the sale of goods. For example, purchasing smart TVs wherein the TV is a tangible good that operates some digital content (e.g., streaming audio).
- $RegD_4$ is an amendment (Law of 25 March 2020) to several finance-related laws in Luxembourg. The amendment establishes a central electronic data retrieval system related to payment and bank accounts for credit institutions in Luxembourg [33].

We had the following criteria in mind while selecting regulatory documents for our study. First, we selected documents that were relevant to software compliance requirements. Second, we focused on documents which Mario was familiar with, since Mario had to spend substantial effort in reviewing them. Third, to draw meaningful conclusions to our empirical evaluation, we were interested in documents that are reasonably large and diverse in terms of the systems to which they are applicable. Considering these criteria, Mario and Luigi elected the above four regulatory documents.

For the purpose of this study, the experts were instructed to propose questions relevant to compliance requirements and highlight the right answers in the regulatory documents. The process started with a half-day training session for Mario and Luigi on automated QA and examples of question-answer pairs from existing datasets (e.g., SQuAD [24]), resulting from previous research on QA for legal domain [4]. Luigi also discussed with Mario the compliance requirements from a requirements engineer's perspective and then presented a few example requirements. After that, both participated in a collaborative exercise, wherein they read through articles from $RegD_1$, identified questions and extracted answers relevant to regulatory compliance. The exercise simulated the manual process of extracting compliance requirements. This exercise lasted approximately a working week, during which the experts identified a total of 36 question-answer pairs from $RegD_1$.

Thereafter and based on the understanding and lessons learnt in the collaborative exercise, a total of 87 question-answer pairs were primarily identified by Mario from $RegD_2 - RegD_4$. These question-answer pairs were generated over a span of three working weeks. Luigi was then instructed to independently review this set of question-answer pairs and reformulate, refine, or propose new questions from the analyzed regulatory documents. Luigi proposed an additional set of 11 question-answer pairs.

The two experts had then another round of validation through joint discussions about the set of 98 question-answer pairs. This round resulted in a set of 33 question-answer pairs from $RegD_2$, and 19 pairs each from $RegD_3$ and $RegD_4$. Overall, they filtered out questions whose answers contained references to other regulatory documents, spanned multiple articles in the same document, or whose answers were too long or not contiguous text in the regulatory document. These decisions were mainly driven by the limitations of the QA models. As discussed in Section II, these QA models are designed to address MRC, i.e., the answer to a given question has to be contiguous text in a text passage.

Despite the above limitations, our work is significant since most questions, in practice, match answers that are in contiguous text. Further, though this remains to be investigated in a systematic fashion, we expect to be able to identify cases with non-contiguous answers by analyzing similarity scores. Such cases should indeed lead to lower and similar similarity scores across multiple context spans. Handling cross-referencing in regulatory documents would have complicated the problem. Hence, finding an answer that requires analyzing multiple regulatory documents is left for future work.

As a result of the process described above, our ground truth consisted of a set of 107 questions which were manually curated and respective answers to these questions. We note that the answers to a few questions could have been potentially found in multiple articles in a regulatory document, due to some degree of repetition in the legal language. For these cases, Mario selected the article that was considered as the most relevant to the question. Our ground truth is composed of an answer for each question and the article where the answer occurs.

D. Evaluation Procedure

We answer our RQs using the experiments below.

EXPI. This experiment addresses RQ1. In EXPI, we evaluate the alternative metrics CS-TFIDF and BCE discussed in step B of our approach in Section III-B for ranking and selecting the top-k context spans. We evaluate these two alternatives using one of the widely-applied metrics in the IR domain, that is Recall@k [34]. Recall@k assesses how often our approach retrieves the context (containing the right answer) for all questions in our ground truth.

For each input question $\mathbf{q}$ and a regulatory document $RegD_i$, we compare a context span c_i selected by our approach among the top-k relevant context spans ($\mathcal{C}_r$) against the article (Art) containing the answer in our ground truth. Following this, we define a *true positive (TP)* when at least one $c_i \in \mathcal{C}_r$ is equal to, or is contained in Art. Recall that a context span in our approach can be an article or part of an article (see step A of our approach discussed in Section III-A). A TP further requires that the answer in the ground truth for $\mathbf{q}$ is subsumed by $c_i \in \mathcal{C}_r$. We define a *false negative (FN)* when no $c_i \in \mathcal{C}_r$ satisfies the two conditions above related to matching p in the ground truth and fully subsuming the answer. We then compute Recall@k as *TP/(TP+FN)* for k=1, 3, 5. Selecting these values for k is motivated by practical considerations. The effort that a requirements engineer needs to find an answer in five context spans is not significant compared to having to go through the entire regulatory document including hundreds of context spans.

EXPII. This experiment addresses RQ2. EXPII compares the accuracy of four QA models in finding the right answer to an input question $\mathbf{q}$, in a given context span. The models namely BERT, ALBERT, RoBERTa, and ELECTRA, are presented in Section II. To draw meaningful conclusions, we evaluate steps B and C of our approach (Fig. 2) independently in EXPI and EXPII. Specifically, we assume that the right context span for $\mathbf{q}$ is given in EXPII. The rationale is to understand how well the QA models fare against one another in extracting the answer to $\mathbf{q}$, irrespective of whether the right context span is accurately selected among the top-k by the approach.

We report accuracy (A), computed as the proportion of questions correctly answered to the total number of questions posed to the model. More specifically, we evaluate the answers extracted by the model, for a given $\mathbf{q}$, against the right answer provided for $\mathbf{q}$ in our ground truth. For each $\mathbf{q}$, an extracted answer by the model is deemed correct if it is contained in or contains the right answer as per the ground truth. The rationale for considering such partial matching (instead of full matching) is that, a partial match still greatly reduces the effort and time for a requirements engineer. Indeed, even with a partial match, it is easy for the engineer to identify the answer, as the rest of the answer is contiguous in the text.

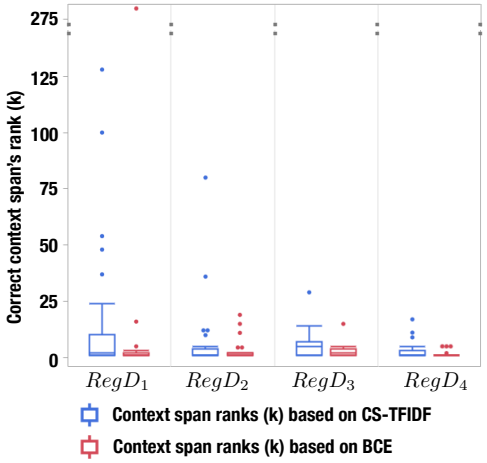

Fig. 3. Box Plots showing correct context span's rank (k) using CS-TFIDF and BCE (**RQ1**.)

TABLE I
RECALL@k: $R@k$ (%) – OF CONTEXT SPAN SELECTION METHODS (**RQ1**)

$RegD$	$\mathcal{C}^\dagger$	Q†	CS-TFIDF			BCE		
			R@1	*R@3*	*R@5*	*R@1*	*R@3*	*R@5*
$RegD_1$	301	36	44.4	66.7	69.4	63.9	91.7	94.4
$RegD_2$	120	33	60.6	69.7	84.8	63.6	84.8	90.9
$RegD_3$	102	19	26.3	36.8	57.9	47.4	63.2	94.7
$RegD_4$	23	19	52.6	78.9	84.2	78.9	84.2	94.7
Total	546	107	47.7	64.5	74.8	78.9	84.2	93.5

† Q is the total number of questions and $\mathcal{C}$ is the total number of context spans generated by our approach from the document.

EXPIII. This experiment addresses RQ3 by running the most accurate QA model (from EXPII) over the best performing text similarity method (from EXPI) on $RegD_1 - RegD_4$. The experiment was conducted on a laptop with 2.3 GHz CPU and 32 GB memory.

E. Results and Discussion

RQ1. Fig. 3 shows box-plots to compare the rank of the correct context span (k) for each question using CS-TFIDF and BCE. Over all four regulatory documents ($RegD_1$ – $RegD_4$), BCE ranks the correct context spans higher than CS-TFIDF, i.e., closer to the top ($k = 1$). A paired Wilcoxon signed-rank test [35] shows statistically significant differences between the two similarity metrics (p-value = 0.0001). Further, the average ranking for BCE is 5 whereas it is 8 for CS-TFIDF. Table I compares the Recall@k results obtained for CS-TFIDF and BCE for k=1, 3, and 5. The table further shows (in the last row) the Recall@k results across all four regulatory documents. For all k values considered in EXPI, BCE significantly

outperforms CS-TFIDF for ranking the context spans and selecting those that are likely to contain the answers to an input question. When accounting for all documents, regarding Recall@k differences for $k = 1$, 3, and 5, BCE outperforms CS-TFIDF with an average percentage point gain of 31.2, 19.7, and 18.7, respectively. Such results suggest practically significant differences. Further analysis using Fisher's exact test [36] on the proportions of relevant context spans ranked at $k = 1$, 3, and 5, shows statistically significant differences between BCE and CS-TFIDF ($p - values < 0.05$).

In practice, a requirements engineer who has a question about system's compliance must read through the regulatory document and find the answer. Retrieving a set of k context spans where the answer is expected to be is highly advantageous to the engineer. As expected, the best results are achieved with $k = 5$ with an overall recall of 93.5%. Though higher k values yield better results, they entail additional effort to review more context spans. For example, considering $k = 10$, BCE yields in our evaluation an overall recall of 94.4%. This increase of 0.9 percentage point (pp for short), when compared to that of $k = 5$, would however result in doubling the effort to read through five additional context spans for each question. In practice, requirements engineers are free to review more context spans, depending on the size of the underlying Reg_D and the time available. This indicates that the requirements engineers need to review five context spans instead of parsing through an entire regulatory document with hundreds of context spans, e.g., $RegD_1$ has 301 context spans.

The answer to RQ1 is that the BERT cross-encoder-based similarity metric (BCE) yields the most accurate context span ranking and selection results. BCE outperforms CS-TFIDF with an average gain in recall of 18.7 pp (at $k = 5$). In 93.5% of the cases (i.e., 100 of 107 questions), BCE selects the right context span among the top five context spans.

We analyzed the *seven* questions where BCE does not highly rank the right context. We observed two patterns in the questions in these cases. The first pattern describes a question which is about a definition of some concept, e.g., "What does [concept X] mean?". The reason is that many context spans in a regulatory document are likely to be relevant for such a question, since it is typical for a concept to be mentioned several times throughout the document. The second pattern describes a question that contains a complicated NL structure. An example question is about the applicability of a directive in certain situations, including composite conditions. Since BCE relies on sentence representations derived from the SBERT model (as discussed in Section II), such cases are confusing to the language model. Thus, the metric does not perform accurately. Further investigation of the questions' patterns is a matter for future work.

TABLE II
ACCURACY (%) OF ANSWER EXTRACTION BY DIFFERENT QA MODELS (**RQ2**).

$RegD$	Q†	BERT	ALBERT	RoBERTa	ELECTRA
$RegD_1$	36	91.7	91.7	**94.4**	**94.4**
$RegD_2$	33	69.7	81.8	**84.8**	**84.8**
$RegD_3$	19	84.2	78.9	**89.5**	78.9
$RegD_4$	19	84.2	**94.7**	**94.7**	73.7
Total	107	82.2	86.9	**90.7**	85.0

† Q is the total number of questions posed on the document.

RQ2. Table II shows the accuracy results obtained in EXPII for the QA models on $RegD_1$ – $RegD_4$. As highlighted in the table, RoBERTa consistently yields the best accuracy in finding the right answers across the regulatory documents when compared to the other QA models. Overall, RoBERTa finds the right answers to more than 90% of the questions, i.e., 97 of 107 questions. Compared to RoBERTa, ELECTRA performs on par on $RegD_1$ and $RegD_2$, but has an average loss of ≈16 pp in accuracy on $RegD_3$ and $RegD_4$. Similarly, ALBERT is as accurate as RoBERTa on $RegD_4$, but performs worse by a margin of ≈4 pp on the remaining three documents.

Given the results of **RQ1**, where ≈94% of the relevant context spans could be selected by our context span selection (step B), the QA models in the subsequent step (i.e., step C) evaluated in RQ2 can answer at most 100 out of 107 questions in our dataset. To give the full picture about how our approach performs, we discuss the results of evaluating steps B and C in sequence, i.e., the performance of the QA models for extracting the answers of the 100 questions whose correct contexts are identified in RQ1. RoBERTa still achieves the best average accuracy. From these 100 questions, RoBERTa correctly answers 91 questions, whereas, BERT, ALBERT and ELECTRA answer 83, 87 and 85 questions, respectively. The higher accuracy of RoBERTa compared to the other models can be attributed to its larger model size and parameter space, as discussed in Section II.

Thus, to answer **RQ2**, we conclude that *RoBERTa is the most accurate QA model for finding the answers to the questions considered in our study, with an average accuracy of ≈91%.*

We further investigated whether the outcome of the models can be understood by a requirements engineer where answers are isolated from the surrounding context, in particular when the engineer is under time pressure. To do so, we manually inspected the resulting answers in isolation from their context spans. We checked how often an answer provides

TABLE III
EXECUTION TIME IN SECONDS FOR STEPS A, B, AND C IN OUR APPROACH (**RQ3**).

	S	$\mathcal{C}$†	Step A	Step B	Step C
$RegD_1$	1501	301	1.0	16.5	21.0
$RegD_2$	480	120	0.8	12.0	20.5
$RegD_3$	389	102	0.6	10.5	22.5
$RegD_4$	111	23	0.5	9.3	20.0
Average	620	137	0.8	12.1	21.0

† S is the total number of sentences and $\mathcal{C}$ is the number of context spans generated by our approach from the document.

a sufficient amount of information concerning the question when not reading the surrounding context. We observed that RoBERTa can provide stand-alone answers in most of the cases. In very few cases, the requirements engineer would still need to revisit the context span for elaborating the highlighted answer.

RQ3. We analyze from a requirements engineer's perspective the scalability of our approach considering results in RQ1 and RQ2, i.e., BCE for selecting the relevant context spans (RQ1) and RoBERTa for extracting the answer (RQ2). Recall our use case where a requirements engineer wants to use our approach to find an answer to a particular question posed on a regulatory document.

For RQ3, we consider all four regulatory documents $RegD_1 - RegD_4$, each with a total number of 1501, 480, 389, and 111 sentences, respectively. We posed, on these four documents, a total of 36, 33, 19, and 19 questions. We report in Table III the average time (in seconds) taken by the three steps in our approach (Fig. 2) for answering a question in each document. To find an answer to a given question, a requirements engineer would be expected to run the following pipeline: (i) preprocessing the question and a regulatory document (step A); (ii) applying BCE to rank and select the top five context spans in the regulatory document (step B); and (iii) running RoBERTa to find the likely answer in each selected context span (step C). Below, we elaborate the execution time for each step of this pipeline.

- **Step A**: Preprocessing $RegD_1 - RegD_4$ requires on average 0.8 seconds. The time needed to preprocess the questions is negligible, since they typically contain limited text.
- **Step B**: Our approach generated an average of 137 context spans across $RegD_1 - RegD_4$. The time needed to compute the text similarity between these context spans and the input questions, using BCE, is on average 12.1 seconds. Ranking the spans (in descending order) and selecting the top five takes negligible time.
- **Step C**: Extracting potential answers to the input questions, in the five selected spans, involves first loading the QA model (RoBERTa in our case), and then running it for a given question on each context span. The times shown in Table III include both loading and running the model. Answering one question requires an average of 21 seconds.

The answer to RQ3 is that execution time of our approach is practical in our application context. Using our approach with BCE and RoBERTa, for answering one question from a regulatory document including an average of 620 sentences, requires a total of ≈34 seconds.

V. THREATS TO VALIDITY

Internal Validity. Learning effect is the main concern for internal validity. In particular, the learning from the underlying QA approach could have potentially biased our experts – Mario and Luigi (see Section IV-C). To mitigate this potential threat, both Mario and Luigi had no exposure to our implementation, and we did not use the automated solution for data collection.

Construct Validity. The answer to a given question can potentially be found in multiple articles in the same regulatory document. However, providing accurate compliance-relevant information to a requirements engineer requires the answer to be embedded in the right context. To ensure that this is properly reflected in our evaluation, we instructed our experts to select the most relevant article for each answer. We further adapted our evaluation metrics to assess the answer not as a stand-alone text but rather within the right context.

External Validity. We evaluated our approach on four regulatory documents of different sizes and for different compliance considerations for a system. Our approach worked well on all documents individually. The consistency seen in our results over these documents provides reasonable confidence in the generalizability of our approach. Further experimentation is nevertheless required to improve external validity, on regulatory documents from other geographical regions, concerning different compliance considerations, and with even larger sizes.

VI. RELATED WORK

This section positions our work against the related work on question-answering in RE and NLP domains.

Question Answering in RE. Question-answering on requirements or other artifacts, to identify requirements, has been studied to some extent in RE. Existing work mostly focuses on querying requirements documents or other textual artifacts for retrieving information related to requirements traceability [37], [38], requirements impacted by a change [39], [40], requirements-related information from online forums [41], and requirements-related information from regulatory documents [4]. Sleimi et al. [4] proposed

an automated querying system using semantic web technologies for taxation laws and is the closest to our approach in the RE literature. The authors tested their approach on eight queries which are relevant to RE activities. They find answers to these queries by extracting relevant semantic metadata, e.g., querying about relevant concepts in a domain and retrieving the metadata *actor*, *location*, *time*. Their approach achieves an average recall of ≈81%. Our work differs from Sleimi et al.'s in two ways, (1) we address questions and find free-text answers that are not restricted to previously-extracted semantic metadata, and (2) we leverage a larger dataset from four European directives jointly created by subject-matter experts from both the legal and RE domains.

Question Answering in NLP. This community has long investigated the topic. Solutions and datasets have been proposed early on in the QA track launched by the Text REtrieval Conference (TREC) [42], [43], [44]. Traditional QA methods proposed in the literature preprocess the input question and extract meta information related to the type of the expected answer [45], [46], [47], [48], [49], [50]. For example, the question that starts with *who* expects an answer of type *person*. The answer type is then used to find the relevant text passage to the input question from a document collection. These approaches address factoid questions, i.e., questions whose answers are facts that can be expressed in short text [10]. Manually annotated datasets generated from news articles are released within TREC over several editions. These datasets are however relatively small and structured around optimizing the answer extraction step in traditional QA methods.

With the recent shift in the NLP landscape, new approaches and larger datasets emerged. Yang et al. [51] introduced the WikiQA dataset, which consists of 3,047 questions generated from Bing search engine query logs. The authors further propose a QA method based on convolutional neural networks. Other datasets, which are currently considered as benchmarks for MRC, include the SQuAD (Stanford Question Answering Datasets). SQuAD 1.1 [24] is a collection of 100K questions-answer pairs manually generated by crowdworkers from the English Wikipedia articles. SQuAD 2.0 [52] extends SQuAD 1.1 with over 50,000 unanswerable questions written by crowdworkers to look similar to answerable ones. The current state-of-the-art for QA utilizes large-scale pre-trained language models for finding the answer of a given question in a text passage (i.e., MRC) [53], [14], [16], [17], [21], [54]. QA for the legal domain has also been studied in the NLP literature. Ravichander et al. [6] present a PrivacyQA dataset consisting of 1750 questions about the privacy policies of mobile applications. The questions are collected through crowdsourcing, whereas the answers are selected by seven experts with legal training. The authors evaluated BERT on their dataset and reported a recall of 35.5%. Other datasets and approaches have been proposed [7], [8], [9]. Existing work in NLP addresses users' needs to understand legal text, or focus to a large extent on privacy-related topics. In contrast, we address in our work a wider spectrum of compliance-related topics and devise our solution to be beneficial for requirements engineers. While a normal reader might ask generic questions to acquire knowledge about some legal topic, a requirements engineer poses legal questions relevant to eliciting a complete set of compliance requirements. Therefore, we consider in our work regulations with a significant impact on the compliance of software systems, e.g., $RegD_2$ is about the supply of digital services.

VII. Conclusion

In this paper, we proposed an automated question-answering (QA) approach for assisting requirements engineers in retrieving compliance-relevant information from regulations. Instead of parsing through hundreds of pages of a regulation, a requirements engineer can simply pose, using our approach, a question in plain NL about information related to the compliance of a system under development. We then identify, using information retrieval-based methods, a set of context spans in the regulation that are relevant to the input question. Our approach further applies large-scale language models (e.g., BERT) to find a likely answer in each context span. We empirically evaluated our approach on 107 question-answer pairs manually curated by subject-matter experts from four European regulations (e.g., the general data protection regulation). Over this dataset, our approach can identify the context span containing the right answer to a given question with an average recall of ≈94% and can extract the correct answer from such context spans with an average accuracy of ≈91%.

In the future, we would like to extend our approach to be able to address questions and answers which involve cross-referencing in a regulation. Furthermore, we plan to conduct user studies involving requirements engineers and/or legal experts to assess the practical usefulness of our approach. Concretely, we aim to compare the time and effort to extract compliance information manually versus relying on the assistance of our approach.

Acknowledgement. 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] P. Klaus and R. Chris, *Requirements Engineering Fundamentals*, 1st ed. Rocky Nook, 2011.

[2] D. E. Leidner and O. Tona, "The care theory of dignity amid personal data digitalization." *MIS Quarterly*, vol. 45, no. 1, 2021.

[3] EU (2019/770), "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," 2019. [Online]. Available: http://data.europa.eu/eli/dir/2019/770/oj

[4] A. Sleimi, M. Ceci, N. Sannier, M. Sabetzadeh, L. Briand, and J. Dann, "A query system for extracting requirements-related information from legal texts," in *27th IEEE International Requirements Engineering Conference*. IEEE, 2019.

[5] A. Sleimi, N. Sannier, M. Sabetzadeh, L. Briand, and J. Dann, "Automated extraction of semantic legal metadata using natural language processing," in *Proceedings of the 26th IEEE International Requirements Engineering Conference*, 2018.

[6] A. Ravichander, A. W. Black, S. Wilson, T. Norton, and N. Sadeh, "Question answering for privacy policies: Combining computational and legal perspectives," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019.

[7] P. Delfino, B. Cuconato, G. Paulino-Passos, G. Zaverucha, and A. Rademaker, "Using OpenWordnet-PT for question answering on legal domain," in *Proceedings of the 9th Global Wordnet Conference*. Global Wordnet Association, 2018.

[8] P. M. Kien, H.-T. Nguyen, N. X. Bach, V. Tran, M. Le Nguyen, and T. M. Phuong, "Answering legal questions by learning neural attentive text representation," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

[9] S. Khazaeli, J. Punuru, C. Morris, S. Sharma, B. Staub, M. Cole, S. Chiu-Webster, and D. Sakalley, "A free format legal question answering system," in *Proceedings of the Natural Legal Language Processing Workshop*, 2021, pp. 107–113.

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

[11] Z. Zhang, H. Zhao, and R. Wang, "Machine reading comprehension: The role of contextualized language models and beyond," *arXiv preprint arXiv:2005.06249*, 2020.

[12] S. Abualhaija, C. Arora, A. Sleimi, and L. Briand, *"Online Annex (online)"*, 2022, available at shorturl.at/ETX17, February 2022.

[13] A. Alexandrescu and K. Kirchhoff, "Factored neural language models," in *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 2006, pp. 1–4.

[14] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[15] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *IEEE International Conference on Computer Vision*, 2015, pp. 19–27.

[16] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *CoRR*, vol. abs/1909.11942, 2019.

[17] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019.

[18] S. Nagel, "Cc-news," 2016. [Online]. Available: https://commoncrawl.org/2016/10/news-dataset-available/

[19] A. Gokaslan and V. Cohen, "Openwebtext corpus," 2019. [Online]. Available: http://Skylion007.github.io/

[20] T. H. Trinh and Q. V. Le, "A simple method for commonsense reasoning," *CoRR*, vol. abs/1806.02847, 2018. [Online]. Available: http://arxiv.org/abs/1806.02847

[21] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: pre-training text encoders as discriminators rather than generators," *CoRR*, vol. abs/2003.10555, 2020.

[22] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *CoRR*, vol. abs/1908.10084, 2019.

[23] C. Zeng, S. Li, Q. Li, J. Hu, and J. Hu, "A survey on machine reading comprehension—tasks, evaluation metrics and benchmark datasets," *Applied Sciences*, vol. 10, no. 21, p. 7640, 2020.

[24] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100, 000+ questions for machine comprehension of text," *CoRR*, vol. abs/1606.05250, 2016.

[25] A. Aizawa, "An information-theoretic perspective of tf–idf measures," *Information Processing & Management*, vol. 39, no. 1, pp. 45–65, 2003.

[26] C. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*, 1st ed. Cambridge University Press, 2008.

[27] E. Loper and S. Bird, "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*, 2002.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[29] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020.

[30] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 2016.

[31] EU (GDPR), "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," 2016. [Online]. Available: http://data.europa.eu/eli/reg/2016/679/oj

[32] EU (2019/771), "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," 2019. [Online]. Available: http://data.europa.eu/eli/dir/2019/771/oj

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

[34] D. Harman, "Information retrieval evaluation," *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 3, no. 2, pp. 1–119, 2011.

[35] D. W. Zimmerman and B. D. Zumbo, "Relative power of the wilcoxon test, the friedman test, and repeated-measures anova on ranks," *The Journal of Experimental Education*, vol. 62, no. 1, pp. 75–86, 1993.

[36] G. J. Upton, "Fisher's exact test," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 155, no. 3, pp. 395–402, 1992.

[37] J. Lin, Y. Liu, J. Guo, J. Cleland-Huang, W. Goss, W. Liu, S. Lohar, N. Monaikul, and A. Rasin, "TiQi: A natural language interface for querying software project data," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2017, pp. 973–977.

[38] N. Niu, W. Wang, A. Gupta, M. Assarandarban, L. Da Xu, J. Savolainen, and J.-R. C. Cheng, "Requirements sociotechnical graphs for managing practitioners' traceability questions," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 4, pp. 1152–1162, 2018.

[39] C. Arora, M. Sabetzadeh, A. Goknil, L. Briand, and F. Zimmer, "NARCIA: an automated tool for change impact analysis in natural language requirements," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 962–965.

[40] ——, "Change impact analysis for natural language requirements: An NLP approach," in *23rd IEEE International Requirements Engineering Conference*, 2015.

[41] G. M. Kanchev, P. K. Murukannaiah, A. K. Chopra, and P. Sawyer, "Canary: an interactive and query-based approach to extract requirements from online forums," in *2017 IEEE 25th International Requirements Engineering Conference*. IEEE, 2017, pp. 470–471.

[42] E. M. Voorhees, "The TREC-8 question answering track report," in *Proceedings of The Eighth Text REtrieval Conference*, vol. 500-246, 1999.

[43] ——, "Overview of the TREC 2001 question answering track," in *Proceedings of The Tenth Text REtrieval Conference*, vol. 500-250, 2001.

[44] S. M. Harabagiu, D. I. Moldovan, C. Clark, M. Bowden, A. Hickl, and P. Wang, "Employing two question answering systems in TREC 2005," in *Proceedings of the Fourteenth Text REtrieval Conference*, vol. 500-266, 2005.

[45] S. M. Harabagiu, M. Pasca, and S. J. Maiorano, "Open-domain textual question answering techniques," *Natural Language Engineering*, vol. 9, pp. 231 – 267, 09 2003.

[46] E. Grois, "Learning strategies for open-domain natural language question answering," in *43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*. The Association for Computer Linguistics, 2005, pp. 85–90.

[47] J. Lin, "The role of information retrieval in answering complex questions," in *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*. The Association for Computer Linguistics, 2006.

[48] D. Demner-Fushman and J. Lin, "Answer extraction, semantic clustering, and extractive summarization for clinical question answering," in *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*. The Association for Computer Linguistics, 2006.

[49] M. H. Heie, E. W. D. Whittaker, and S. Furui, "Optimizing question answering accuracy by maximizing log-likelihood," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics, 2010, pp. 236–240.

[50] N. Duan, "Minimum bayes risk based answer re-ranking for question answering," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics, 2013, pp. 424–428.

[51] Y. Yang, W.-t. Yih, and C. Meek, "WikiQA: A challenge dataset for open-domain question answering," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 2013–2018.

[52] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for SQuAD," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2018, pp. 784–789.

[53] M. Hu, Y. Peng, F. Wei, Z. Huang, D. Li, N. Yang, and M. Zhou, "Attention-guided answer distillation for machine reading comprehension," in *Proceedings of Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018, pp. 2077–2086.

[54] O. Khattab, C. Potts, and M. Zaharia, "Relevance-guided supervision for OpenQA with ColBERT," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 929–944, 2021.