Cross-Policy Compliance Detection via Question Answering

Marzieh Saeidi[1], Majid Yazdani[1], and Andreas Vlachos[1,2]

[1]Facebook AI
[2]University of Cambridge
{marzieh,myazdani}@fb.com, av308@cam.ac.uk

Abstract

Policy compliance detection is the task of ensuring that a scenario conforms to a policy (e.g. a claim is valid according to government rules or a post in an online platform conforms to community guidelines). This task has been previously instantiated as a form of textual entailment, which results in poor accuracy due to the complexity of the policies. In this paper we propose to address policy compliance detection via decomposing it into question answering, where questions check whether the conditions stated in the policy apply to the scenario, and an expression tree combines the answers to obtain the label. Despite the initial upfront annotation cost, we demonstrate that this approach results in better accuracy, especially in the cross-policy setup where the policies during testing are unseen in training. In addition, it allows us to use existing question answering models pre-trained on existing large datasets. Finally, it explicitly identifies the information missing from a scenario in case policy compliance cannot be determined. We conduct our experiments using a recent dataset consisting of government policies, which we augment with expert annotations and find that the cost of annotating question answering decomposition is largely offset by improved inter-annotator agreement and speed.

1 Introduction

Policy compliance detection is the task of ensuring that a scenario conforms to a policy. It is a task that occurs in various contexts, e.g. in ensuring correct application of government policies (Saeidi et al., 2018; Holzenberger et al., 2020), enforcing community guidelines on social media platforms (Waseem and Hovy, 2016) or the resolution of legal cases (Zhong et al., 2020), etc.

While the task can be modelled as a text-pair classification similar to textual entailment (Dagan et al., 2009; Bowman et al., 2015), i.e. whether a scenario complies with a certain policy, this often results in poor accuracy due to the complexity of the policy descriptions, which often contain multiple clauses connected with each other (Holzenberger et al., 2020). Moreover, even though some types of policy compliance detection can be tackled via supervised classification, e.g. recognizing hateful memes (Kiela et al., 2020) and other forms of hate speech or abusive language on social platforms (Davidson et al., 2017), this requires substantial amounts of training data for one specific policy which is expensive and time-consuming. Furthermore, the requirement for policy-specific training data renders supervised classification difficult to adapt when policies change, e.g. customs rules changing with Brexit, novel forms of hate speech in the context of the Covid pandemic, etc.

In this paper we propose to address policy com-

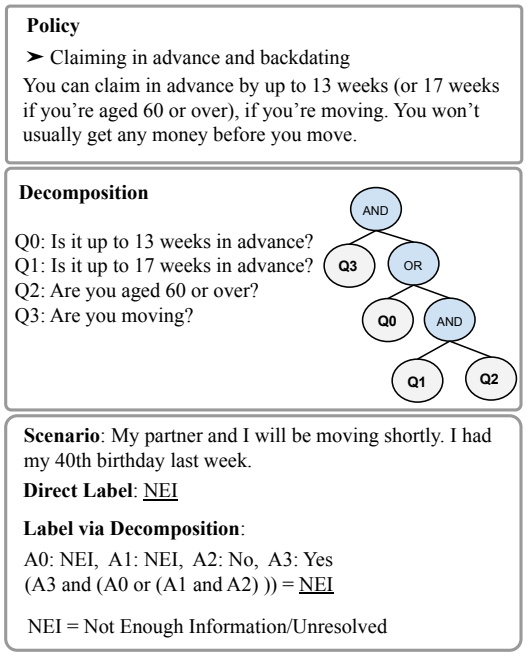

Figure 1: A policy with its decomposition (questions and expression tree), scenario and label.[1]

[1]Policy source can be found here https://www.gov.uk/housing-benefit/how-to-claim

pliance detection via decomposing it to question answering (QA) and an expression tree. Each policy description is converted into a set of questions whose answers are combined in a logical form represented by the expression tree to produce the label. This allows for better handling of complex policies, and to take advantage of the large-scale datasets and models developed for question answering (Rajpurkar et al., 2016; Clark et al., 2019). In addition, the questioning of the scenario with respect to parts of the policy identifies which parts were used in order to determine compliance and which information is missing in case the scenario could not be resolved. This in turn, enhances the interpretability of the model's decisions and can identify areas for improvement. Figure 1 shows an example application of our approach in the context of a government policy where the missing date of the move in the scenario results in its compliance with the policy being unresolved.

We demonstrate the benefits of policy compliance detection via QA using a dataset that contains policies, decomposition questions and expression trees and scenarios. While the policies and scenarios in the dataset are taken from ShARC (Saeidi et al., 2018), we augment them with expression trees and answers to each question for all the scenarios and policies to create the Question Answering for Policy Compliance (QA4PC) dataset. The results of our experiments demonstrate that we can achieve an accuracy of 0.69 for policies unseen during the training (an increase of 0.25 absolute points over the entailment approach) and an accuracy of 0.59 (an increase of 0.22 absolute points) when no in-domain data is available for training. We also show that our approach is more robust compared to entailment when faced with policies of increasing complexity. In addition, we find that the cost of annotating question answering decomposition and expression trees is largely offset by improved inter-annotator agreement and speed. Finally, we release the QA4PC dataset to the community to facilitate further research[2].

2 Previous Work on Policy Compliance

Policy compliance as an entailment task has been studied by Holzenberger et al. (2020) and referred to as statutory reasoning entailment. They find that the application of machine reading models exhibits low performance, whether or not they have been fine-tuned to the legal domain, due to the complexity of the policies. As an alternative approach, they propose a rule-based system. While their work focuses solely on the US Tax Law, the dataset we use in our experiments (Saeidi et al., 2018) contains a wide range of policies from different countries. Furthermore, the dataset used in (Holzenberger et al., 2020) only contains scenarios that are either compliant or not compliant with a policy. However, in real world use, systems should be capable of recognizing that a decision cannot be made in cases there is information missing from the scenario.

ShARC (Saeidi et al., 2018) is a dataset for conversational question answering based on government policies. In ShARC, the task is to predict what a system should output given a policy and a conversation history, which can be an answer or a follow-up clarification question. The task is evaluated in an end-to-end fashion, measuring the system's ability to provide the correct conversation utterance, However, policy compliance detection is neither investigated nor evaluated.

Similarity search or matching has been used to match a new post to known violating posts on social media such as hoaxes, an objectionable video or image, or a hateful meme (Ferreira and Vlachos, 2016; Wang et al., 2018; fbm). For textual content, this can be compared to an entailment task. This approach requires a bank of existing violating content for each policy. By using the policy description and decomposing it into several QA tasks, breaches for new policies can be detected, as we show in our experiments.

Much work has focused on learning important elements of privacy policies to assist users in understanding the terms of the policies devised by different websites. Shvartzshanider et al. (2018) uses question answering to extract different elements of privacy policies that are informative to the users. Nejad et al. (2020) and Mustapha et al. (2020) assign pre-defined categories to privacy policy paragraphs using supervised classification. While these works aim to help users in understanding complex text of privacy policies, they do not aim to identify compliance and they mainly focus on privacy policies. In our work, we look at a wide range of government policies, for different governments and focus on an approach for cross-policy compliance detection.

[2]Please contact the authors to access the dataset.

3 Policy Compliance Detection via QA

In this section we describe our proposed approach for policy compliance detection via question answering. First we formulate the task as textual entailment following previous work Holzenberger et al. (2020). Then we describe how our proposed expression trees combined with QA operate. Finally we discuss the implications of our approach for training and data annotation requirements.

3.1 Policy Compliance Detection (PCD)

We de ne the task of policy compliance detection (PCD) as deciding whether a scenario is compliant with a policy. For a policy and a scenario, the task is to provide a label in [Yes, No, Not Enough Information]. We use NEI in short for Not Enough Information for the remaining of the paper. For an input $(p; s)$, the output is the label l. See Figure 1 for an example. If framed as textual entailment, a policy is the premise and the scenario is the hypothesis, as there are typically multiple scenarios per policy. The labels [Yes, No, NEI] correspond to [entailment, contradiction, neutral].

3.2 Question Answering for PCD

Our proposed decomposition of policy compliance detection into question answering infers the label using the answers to the questions and the expression tree. More formally, the QA model will take an input $(s; q_i)_{i=1:::K\ \ 1}$ and output $a_i \ 2$ [Yes, No, NEI], where K is the number of questions representing the policy p. The expression tree combines the answers to the questions $fa_i g_{i=0:::K\ \ 1}$, using a logical expression based on the rules in the policy description to infer the nal label l of whether a scenario is in compliance with the policy or not. Figure 1 shows an example of a policy decomposed into questions and an expression tree and how it is applied to a scenario to infer a label.

An expression tree can contain OR, AND and NOT operators. Expression trees can be evaluated in the same way that logical expressions are evaluated where Yes is considered as True and No as False. Similar to logical expressions, we can evaluate an expression tree even when the answers to some of the questions are NEI. For example, the inferred label for "Q0 AND Q1" is False if the answer to Q0 is False and the answer to Q1 is NEI. If the answer to Q0 is True and the answer to Q1 is NEI, the inferred label is NEI. It is worth noting that we do not always need to correctly predict the answers to all the questions in the expression tree in order to infer the correct label. For example, for the expression tree in Figure 1, if the answer to Q3 is correctly predicted as No, the model doesn't need to make correct predictions for Q0, Q1 or Q2.

The underlying QA task is similar to BoolQ (Clark et al., 2019) with the exception that an answer can be NEI. This can be compared to SQUAD 2.0 (Rajpurkar et al., 2018) where the corpus contains unanswerable questions, but the answers are spans extracted from the passage instead of boolean. In this work, we assume that the expression trees are provided and leave the task of inferring them for future work.

3.3 Training and Data Requirements

Expression trees are not required during the training of our approach; they are only used during the testing to infer the label for the scenario given the policy. For this reason, we only provide the annotation of expression trees for the data that will be used for evaluation (dev and test sets) to demonstrate the bene ts of our proposed approach. Using QA as the subtask, in which we decompose PCD into, enables us to apply our approach to policies unseen in training (a.k.a. cross policy setup), since QA models are typically developed for and evaluated on their generalization ability to questions unseen in their training data.

4 Data

In this section we discuss the creation of the question-answering for policy detection (QA4PC) dataset which we introduce in this work to support our experiments. We create QA4PC using the policies and the scenarios from the publicly available training and development splits of existing dataset ShARC (Saeidi et al., 2018). A scenario in ShARC is a real-world situation described by a user who is conversing with a system to nd out whether they comply with a policy. Each conversation utterance in ShARC has a policy, a question and an answer. An utterance may include a conversation history (a list of QAs) and/or a scenario which is built from a conversation list (a list of questions and answers). The answer to an utterance can be Yes/No or a follow-up clari cation question.[3]

However, while ShARC conversations contain some of the questions related to the policies, we

[3]More details on ShARC data can be found here: https://sharc-data.github.io/data.html

found that many were missing, as the conversation progress may render answers to them irrelevant. A such example is provided in Figure 6 in the Appendix. Details on number of QAs per policy for both datasets are discussed in section 4.4.

As neither entailment-style classification nor the question answering expression trees require complete question-answer sets per instance or expression trees for training, we convert 70% of ShARC policies for training of entailment and QA tasks (see section 4.1 for more details). The procedure for converting the data is explained in the next section. In order to ensure the evaluation datasets (dev and test sets) are correctly annotated, we augment the remaining 30% of ShARC policies with questions and expression trees, and the scenarios with the entailment labels and answers to all questions of the related policy (see Section 4.2). Two annotators, co-authors of this paper and UK nationals, were involved in the annotation described in this section.

4.1 Converting ShARC to PCD

Entailment Instances We take utterances that have non-empty scenarios. The entailment answer to the scenario is Yes/No if the answer to the conversation utterance in ShARC is Yes/No and the conversation history is empty. Conversation history refers to a list of questions and answers that has already been exchanged between an agent and a user. Otherwise, the assigned label is NEI. This is because if the conversation history is not empty or the final answer is a follow up question, some of the necessary information related to the policy is not mentioned in the scenario and is acquired from history or will be answered by the user in the next step of the conversation in the form of an answer to the follow up question.

QA Instances We construct the set of unique follow-up questions related to each policy over all the scenarios as the set of questions for that policy. If any question from this set appears in the conversation corresponding to a scenario (a list of QAs), we create a QA instance where the passage is the scenario, and the answer is the answer to that question in the QA list. For all the other questions in this set, we add a QA instance using the scenario and the answer NEI. More details about the ShARC dataset and its processing are included in the Appendix in section A.1.

In-Domain Supervision We call the converted ShARC dataset ShARC-PC. There are 482 policies, 4,576 scenarios and 10,398 QAs in the ShARC training set. The policies used in ShARC-PC are distinct from those used in QA4PC. Note that the labels in ShARC-PC are not always accurate because they are assigned using a heuristic, thus even though it is in-domain as it is derived from conversation about government policies, it is noisy. This was specially noted when augmenting the ShARC data to obtain QA4PC.

4.2 Expression Tree Annotation

Annotation Procedure Each policy is decomposed by the annotators into a set of questions and an expression tree based on the rules in the policy description. Each question is assigned an ID (e.g. Q0, Q1) which is used in the expression tree. For each policy, some of the questions can be extracted from the ShARC dataset as mentioned in the previous section. These questions are provided to the annotators. An annotator can change the question or remove it. Annotators can also add questions to expression tree if the existing questions are not sufficient. The UI for this annotation task is depicted in Figure 7 in the appendix. It took the annotators an average of 113 seconds to annotate each tree.

Quality Control Annotators were asked to mark difficult instances and also take note during the annotation if they were in doubt about an instance. After the first round of annotation, the annotators discussed their notes and clarified the guidelines. Finally, each annotator reviewed the other annotator's expression trees and made updates if necessary based on the interim discussion. To further ensure the quality of the expression trees, we asked a third annotator (third co-author, not involved in the annotation otherwise) to review a sample of 50 trees. These trees were selected proportional to their complexity, i.e. number of questions and logical operators. Out of 50 trees, the third annotator did not agree with 4 annotations, i.e. an agreement of 92%. The annotator marked 6 additional instances where they believed the framing of questions can be improved.

4.3 Entailment and QA Annotations

Annotation Procedure Each scenario is paired with its corresponding policy. An annotator is required to choose a label from (Yes/No/NEI) with regards to the policy compliance (i.e. entailment

Figure 2: Label/answer types for training and evaluation datasets for entailment (left) and QA (right) tasks.

	Policies	Scenarios	QAs	Avg QA/Policy
Dev	60	437	1,600	2.31
Test	133	1,099	3,492	2.30

Table 1: Number of instances in QA4PC dataset.

instance). In the same annotation task, the annotators are asked to provide answers to each individual question in the expression tree of the policy (QA instances). The UI for this task is shown in Figure 8 in the appendix.

Quality Control We compared the inferred labels to all scenarios using the expression tree of the corresponding policy against the entailment label provided as a sanity check. Annotators were also asked to take notes during the annotation if needed, and discussed their notes and the discrepancies identified by the sanity check. In some cases, answers were adjusted appropriately. In other cases, the expression tree was updated, and the annotators ensured that the question and answers were updated accordingly.

4.4 QA4PC Dataset

In this section, we present statistics of the QA4PC. As neither our approach nor the entailment baseline require training data annotated with expression trees, we divide the dataset only into dev and test sets. There are 193 policies in QA4PC across both test and dev sets. Each policy has between 1 and 9 questions and between 2 and 55 scenario instances. Table 1 shows more properties of dev and test sets such as the total number of scenarios and individual QAs and average number of QAs per policy. The published dataset contains the training, dev and test sets.

Policies in QA4PC have an average number of 2.31 QAs. This is higher than the average number of QAs per policy in ShARC-PC which is 1.73, because annotators added additional questions to the policies during the annotation when required. Figure 2 shows the distribution of entailment labels and answers to the QA instances between training (i.e. ShARC-PC) and evaluation (i.e. QA4PC) sets. As the figure shows, the training data has a higher number of instances where the entailment labels (left) or answers to the questions are NEI (right).

5 Experimental Setup

Model We use RoBERTa (Liu et al., 2019) for both the entailment baseline and the QA subtask and learn three-way classifiers for each. We used the implementation in the huggingface (Wolf et al., 2020) library. We also experimented with a T5 model (Raffel et al., 2019) on the dev set. Since the results of T5 model were very similar to RoBERTa, we report the performance on all the tasks using RoBERTa.

Entailment We use a cross-encoder set up for the task. For each instance, we combine the policy, scenario and the question using the following format: "premise: [policy] SEP hypothesis: [scenario] CLS". The embeddings of the token CLS is used to classify the instance. This is a common approach for entailment modelling using pre-trained encoders (Devlin et al., 2018; Raffel et al., 2019; Holzenberger et al., 2020).

Question Answering for PCD In the QA approach to PCD, we decompose the task into a QA subtask and expression trees, where we use the expression tree of a policy to combine the answers to the questions in the policy to infer the label. A QA model is trained on QA instances of the dataset. For each QA instance, we combine the scenario and the question using the format: "passage: [scenario] SEP question: [question] CLS". This is a common formulation of QA tasks (Raffel et al., 2019) using transformer models. The embedding of the CLS token is used to assign a label to each instance.

	Avg over Scenarios						Avg over Policies					
Training data	ShARC-PC		BoolQ		SNLI		ShARC-PC		BoolQ		SNLI	
	Test Set											
Entailment	0.44	0.02	NA		0.37	0.02	0.44	0.03	NA		0.42	0.02
QA ET	0.69	0.01	0.59	0.02	NA		0.69	0.02	0.62	0.02	NA	
	Dev Set											
Entailment	0.44	0.01	NA		0.35	0.02	0.48	0.01	NA		0.39	0.02
QA ET	0.70	0.01	0.65	0.02	NA		0.74	0.02	0.65		NA	

Table 2: Cross-policy accuracy averaged on all scenarios and per policy on test and dev sets in terms of macro-accuracy, averaged over 5 runs with varying seeds. QA ET refers to QA decomposition with Expression Trees.

Out-of-Domain Training Data We use the BoolQ (Clark et al., 2019) dataset for training the QA model. BoolQ only contains instances (passage and question pair) with Yes/No answers. We pair random passages and questions to create instances with NEI answers. To train the entailment model, we use the SNLI dataset (Bowman et al., 2015). To adapt the labels to our task, we convert entailment to Yes, contradiction to No and neutral to NEI.

Evaluation Metric We use macro-accuracy (averaged over 3 classes of PCD labels/QA answers) averaged over scenarios, since the labels and answers are not balanced in our dataset (see Figure 2). As some policies have more scenarios than others, we also report results averaged over policies, where we first calculate the macro-accuracy per policy and then average over policies.

Hyperparameters We did a manual tuning of hyperparameters using the dev set. Batch size was set to 16, learning rate to 5e 5, Adam epsilon to 1e 8 and maximum gradient norm to 1.0. We trained all the models for a maximum of 5 epochs with early stopping using the loss on the dev set. We run each experiment 5 times with different random seeds and report the mean and variance. The models were ran on a machine with 8 Tesla V100-SXM2 GPUs, each with 16 GB memory. Each epoch of model training takes about 25 seconds for the entailment baseline and 35 seconds for the QA subtask on ShARC-PC. Number of trainable parameters of the model is 124,647,939.

6 Results

6.1 Model Accuracy

The accuracies of different approaches in detecting policy compliance and using different training data are presented in Table 2.

The first column of the table shows the performance of both approaches in detecting policy compliance on unseen policies when trained on data from ShARC-PC. The accuracy of the entailment approach is 0.44 while the QA decomposition approach reaches an accuracy of 0.69. Table 3 shows the performance of the QA subtask (macro-accuracy over QA instances) on QA4PC test set using in-domain (i.e. ShARC-PC) and out-of-domain (i.e. BoolQ) training data. As mentioned previously, we do not need to answer all the questions in a policy correctly in order to get the correct final label for PCD. For example, in Figure 1, if the answer to Q3 is correctly predicted as No, inferring a correct final label is independent of the remaining answers. This is the reason that a QA model with an accuracy of 0.68 can still achieve an accuracy of 0.69 on the PCD task.

	ShARC-PC		BoolQ	
QA	0.68	0.02	0.59	0.01

Table 3: The performance of the QA task where models are trained on in-domain and out-of-domain data and evaluated on the test set of QA4PC.

	Yes		No		NEI	
QA	0.84	0.03	0.54	0.05	0.66	0.03
PCD	0.80	0.03	0.68	0.04	0.59	0.02

Table 4: The performance of QA and PCD tasks per label using the model trained on ShARC-PC.

Since we have a class imbalance in our dataset, we show the accuracy of the model per label for both the QA and the PCD tasks in Table 4. The most difficult answer for the QA model to predict is No, while the answer with the lowest prediction accuracy for the PCD task is NEI.

Transfer Learning We define transfer learning as learning from a training set that is out-of-domain (i.e. not government policies). Table 2 shows transfer learning results for the PCD task based on training on BoolQ for QA-based approach and SNLI for the entailment baseline. As we can see, the decomposition approach achieves an accuracy of 0.59 when trained on BoolQ data. On the other

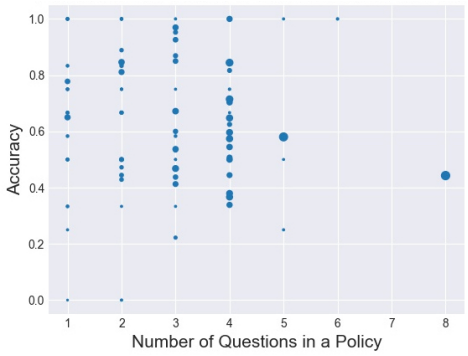

Figure 3: Performance (test set) per policy vs. the complexity of the policy, i.e. the number of questions in the expression tree.

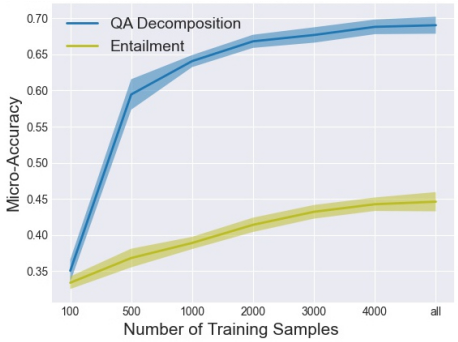

Figure 4: Performance of two approaches on the test set as the number of training samples (scenario) from in-domain data increases.

hand, the accuracy of the entailment model trained on SNLI is 0.37 which is only slightly better than a random system. This is likely due to entailment detection between policy and scenario being a more difficult task due to the complexity of the policy description, and it is not possible to obtain good accuracy without in-domain data.

Performance per Policy We evaluate the accuracy of both approaches averaged over policies to show how well we can infer compliance with a new policy. The results averaged over policies are shown in the right side of Table 2. The accuracies averaged over policies are similar or slightly higher than those averages over all the scenarios. This indicates that there are policies that are difficult to do inference on which have a high number of instances. This is evident for both the entailment and QA decomposition approaches.

Figure 3 shows the average performance of the PDC task using QA decomposition approach for individual policies in the test set of QA4PC versus the complexity of the policy description, i.e. number of questions in the policy expression tree. The size of the circles indicate the number of examples in the dataset for a policy. We used Kendall tau to find the correlation between the number of questions in a policy and the accuracy. Tau coefficient between accuracy and number of questions in policies using QA and entailment approaches are 0.12 (p-value of 0.05) and 0.17 (p-value of 0.01) respectively. This shows that the accuracy through the entailment approach is more negatively affected as the complexity of a policy increases. Finally, it is worth noting that a policy with 8 questions has a high number of instances. The average accuracy over these instances is 0.42 which contributes to the slightly lower performance when averaged over the scenarios reported in Table 2.

Limited Supervision In Figure 4, we show the performance of the model on QA4PC as the amount of in-domain training data from ShARC-PC increases. The results are shown in blue and green for the QA decomposition and the entailment approach respectively. As figure shows, with only 500 scenario instances, we achieve an accuracy of 0.60 through the QA decomposition approach. The entailment approach reaches an accuracy of 0.40 using more than 1000 instances. Note that the number of training instances are based on the number of scenario instances which can include a higher number of QA instances. However we will demonstrate the annotation time is not higher for more QA instances in the next section. Also, it takes only an average of 113 seconds to annotate the expression tree (see section 4.2) which is done only once per policy and used for multiple scenarios.

6.2 Annotation Accuracy and Efficiency

To perform policy compliance detection via QA decomposition, not only we need annotations for expression trees, we also need to annotate the answers to all the questions of a policy for a related scenario instance. Therefore, for each scenario, instead of needing one entailment label, we need annotators to provide answers to all the questions in the expression tree of a policy. One may argue that this approach is more costly as a higher number of annotations is required and that this effort could be used instead to annotate more data on the entailment level that can lead to equally good or better performance. In this section, we describe our experiment to demonstrate the effectiveness of the annotation for the QA decomposition approach in

comparison to the entailment.

Task Setup 50 policies were selected from QA4PC. We include policies with different number of questions (and therefore complexity) in this experiment. For each policy, one scenario is selected randomly from the dataset. Policies are unique to ensure that the accuracy and speed of annotators are not affected by their familiarity with the policies. We then create two annotation tasks for each scenario-policy instance. In the first task, the annotators are presented with the policy and the scenario and they are required to choose one entailment label. In the second task, the scenario and *all* the questions related to the policy are presented to the annotators. An annotator is required to provide an answer to each of the questions.

Agreement and Speed We used two annotators (unpaid volunteers who are not authors of the paper but are native English speakers) to perform both annotation tasks. The average time to provide an entailment label for a policy and a scenario is 39 seconds across two annotators. The average time to answer *all* the questions of a policy with respect to a scenario is 27 seconds. The annotators highlighted the difficulty of annotating the entailment task because they had to create a mental breakdown of a policy while they found the QA task easier to perform as the policy is already decomposed into individual questions. The Cohen's Kappa for entailment agreement between the annotators is 0.38 and for individual questions 0.49. The agreement on the inferred labels using the expression trees based on the answers to the questions is 0.70. This agreement is higher than the agreement for the QA annotation, because as we discussed in section 6.1, not all the the questions need to be answered correctly to infer the correct final label. The results of this annotation experiment shows that annotating the QA labels has lower cost (faster) compared to annotating the overall entailment label, even though an annotator needs to complete more annotations.

7 Discussion

Assumptions In this work, we assume that we can decompose policy descriptions into independent questions that can be executed in parallel. However, designing independent questions is not always straightforward. E.g. it could be useful to have a question in an expression tree that is a follow-up on another question and therefore a QA model will have to be executed sequentially taking into account the earlier question in formulating the follow up one.

Potential for Automatic Generation of Expression Trees In our work, we assume that the expression trees are provided. Providing an expression tree for a policy by those who create it is unlikely to be a substantial overhead, and it can lead to better policy definitions. In addition, in the case of policies on sensitive issues such as hate speech, it might be undesirable to have the expression tree and questions inferred automatically. Moreover, an expression tree for a policy is created only once but used multiple times. As discussed in the results section, the cost of creating expression trees when using the QA approach is offset by the gains in performance compare to the entailment approach.

Nevertheless, on-boarding all the policies of an existing organisation can be time-consuming and can benefit from automation, and with human-in-the-loop approaches we can ensure the accuracy of the trees. Finally, a potential benefit of automating the generation of expression trees is that it can be done jointly with the task of policy compliance detection, such that we generate trees that result in a higher accuracy on the downstream task.

Use of Task Descriptions The use of task descriptions was recently studied in Weller et al. (2020) to answer questions based on a passage. Our work is similar to this work since we use policy descriptions to learn PCD and conduct evaluation in a cross-policy setting. While they answer questions from passage descriptions, we focus on learning policy compliance given a policy and a scenario. Description of relations has been used in (Obamuyide and Vlachos, 2018) to perform zero-shot relation classification.

Question Decomposition Answering complex questions has been a long-standing challenge in natural language processing. Many researchers explored decomposing questions into simpler questions (Wolfson et al., 2020; Min et al., 2019; Ferrucci et al., 2010; Perez et al., 2020). In question decomposition, the objective is to convert a complex question into a list of inter-related sub-questions. While in question decomposition questions are generated automatically, in our work, we consider them given. On the other hand, the answers to the decomposed questions are combined in a more complex manner using an expression tree with logical oper-

ators, while in these works there is no attempt to combine the answers.

8 Conclusion

In this work, we propose to address the task of policy compliance detection via decomposing a policy description into an expression tree. The expression tree consists of a set of questions and a logical expression that combines the questions to infer a final label. Our experiments show that compared to the existing entailment approach (Holzenberger et al., 2020), QA decomposition results in a better model accuracy in a cross-policy setting using in-domain (0:69 vs. 0:44) and out-of-domain (0:59 vs. 0:37) training data. Furthermore, we show that while there is an upfront cost of annotation for expression trees, the cost of QA decomposition annotations is lower than the cost of annotating the entailment task while reaching a higher agreement.

Future work can investigate ways to generate expression trees for policy descriptions automatically. Further, it will be beneficial to demonstrate that this approach is suitable in detecting policy compliance in other domains such as community standards implemented by social media platforms which requires annotation of relevant datasets.

9 Broader Impact Statement

Tools based on our method could potentially improve automated policy enforcement efforts and could also help individuals understand how certain policies may apply to their circumstances. The proposed method offer better interpretability and performance when little data in available. This can provide some insights on where a model may have made a mistake.

We use an existing transformer-based model to show the effectiveness of our method. Transformer-based models are known not to be computationally effective. However, we use a pre-trained model and fine-tune it on a relatively small dataset.

References

Open-sourcing photo- and video-matching technology to make the internet safer. https://about.fb.com/news/2019/08/open-source-photo-video-matching/. Accessed: 2020-05-10.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. Dataset License: CC BY-SA 3.0 https://creativecommons.org/licenses/by-sa/3.0/.

I. Dagan, B. Dolan, B. Magnini, and D. Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

Nils Holzenberger, Andrew Blair-Stanek, and Benjamin Van Durme. 2020. A dataset for statutory reasoning in tax law entailment and question answering. In *Proceedings of the 2020 Natural Legal Language Processing (NLLP) Workshop, 24 August 2020, San Diego, US*.

Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. 2020. The hateful memes challenge: Detecting hate speech in multimodal memes. *arXiv preprint arXiv:2005.04790*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. *arXiv preprint arXiv:1906.02916*.

Majd Mustapha, Katsiaryna Krasnashchok, Anas Al Bassit, and Sabri Skhiri. 2020. Privacy policy classification with xlnet (short paper). In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 250–257. Springer.

Najmeh Mousavi Nejad, Pablo Jabat, Rostislav Nedelchev, Simon Scerri, and Damien Graux. 2020. Establishing a strong baseline for privacy policy classification. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 370–383. Springer.

Abiola Obamuyide and Andreas Vlachos. 2018. Zero-shot relation classification as textual entailment. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 72–78.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. *arXiv preprint arXiv:2002.09758*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. *arXiv preprint arXiv:1809.01494*.

Yan Shvartzshanider, Ananth Balashankar, Thomas Wies, and Lakshminarayanan Subramanian. 2018. Recipe: Applying open domain question answering to privacy policies. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 71–77.

Xuezhi Wang, Cong Yu, Simon Baumgartner, and Flip Korn. 2018. Relevant document discovery for fact-checking articles. In *Companion Proceedings of the The Web Conference 2018*, pages 525–533.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E Peters. 2020. Learning from task descriptions. *arXiv preprint arXiv:2011.08115*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. 2020. Jec-qa: A legal-domain question answering dataset. In *Proceedings of AAAI*.