Towards an Automated Extraction of ABAC Constraints from Natural Language Policies

Manar Alohaly, Hassan Takabi, Eduardo Blanco

Towards an Automated Extraction of ABAC Constraints from Natural Language Policies

Manar Alohaly[1,2](✉), Hassan Takabi[1], and Eduardo Blanco[1]

[1] University of North Texas, Denton, TX, USA
ManarAlohaly@my.unt.edu
{Takabi,Eduardo.blanco}@unt.edu
[2] Princess Nourah bint Abdulrahman University, Riyadh, KSA

Abstract. Due to the recent trend towards attribute-based access control (ABAC), several studies have proposed constraints specification languages for ABAC. These formal languages enable security architects to express constraints in a precise mathematical notation. However, since manually formulating constraints involves analyzing multiple natural language policy documents in order to infer constraints-relevant information, constraints specification becomes a repetitive, time-consuming and error-prone task. To bridge the gap between the natural language expression of constraints and formal representations, we propose an automated framework to infer elements forming ABAC constraints from natural language policies. Our proposed approach is built upon recent advancements in natural language processing, specifically, sequence labeling. The experiments, using Bidirectional Long-Short Term Memory (BiLSTM), achieved an F1 score of 0.91 in detecting at least 75% of each constraint expression. The results suggest that the proposed approach holds promise for enabling this automation.

Keywords: Access Control Policy · Attribute-based Access Control · Natural Language Processing · Constraints Specifications

1 Introduction

Attribute-based access control (ABAC) is an effective model in governing the controlled access to information and resources using the notion of rules. Authorization rules are defined by means of attributes which describe properties or characteristics of entities involved in authorization decisions. Hence, a user may exercise a permission only if the assigned values of the attributes of the user and object satisfy the configuration of the authorization rule [8]. Further, to comply with high-level organizational policies, these attributes often need to be constrained. Hence, a proper selection and specification of the constraints are crucially important for policy adherence.

Recently, several studies have focused on the formal specification and enforcement of constraints in the context of ABAC systems [2, 3, 10, 21]. However, defining formal constraints requires a significant amount of training and mathematical sophistication. It also involves a considerable degree of expert manual

effort. That is, the security architects of the system have to analyze multiple natural language policy documents to extract constraints-relevant information necessary to provide the formal notion of constraints. Therefore, the effectiveness of the set of formally defined constraints is heavily dependent on the individual's particular level of skill. This fact introduces a need for an automated aid to assist security architects in analyzing natural language policy documents and defining the formal constraints.

To that effect, we present this work, that builds upon the state-of-the-art advances in natural language processing (NLP), in order to aid security architects in extracting information necessary to form ABAC constraints from natural language policies. This automation does not only aid security architects in the constraints specification task, but also enables the traceability between formal constraints expressions and the policies they represent. This in turn, can be used to defend against intentional or unintentional unauthorized attributes assignments when manually handled by the security architect. To the best of our knowledge, there is no work in the literature that addresses the automated extraction of constraints directly from natural language policies.

The main research contributions of this paper are as follows: (1) we design an annotation scheme to capture ABAC constraint expressions within natural language access control policies (NLACPs), (2) we design a framework to extract the constraints from NLACPs, and (3) we present experimental results showing that constraints extraction task can be automated reliably. The remainder of this paper is organized as follows: Section 2 provides the background information. In Section 3, we discuss our approach to automate the extraction of ABAC constraints from natural language policies. Our dataset and experimental results are discussed in Section 4. We review the related work in Section 5; and, in Section 6, we conclude our study with the recommendations for future work.

2 Background

In the following subsections, we provide background information regarding: (1) ABAC-based constraints specification language that we adopt to encode constraints, (2) natural language parsing strategies and (3) sequence labeling model.

2.1 Overview of Constraints Specifications in ABAC

Since authorization decisions in ABAC are based on the notion of attributes, the constraints are also defined according to the relationships among these attributes. Particularly, constraints in ABAC can be seen as the component that defines what values can or cannot be assigned to attributes of entities involved in the access. For instance, suppose a university system has defined a policy as "*An adjunct faculty member should not be considered a full-time employee.*" Two attributes, possibly named position and employmentType, were used in this system to express the faculty position and employment type information. In this case, a mutual exclusion constraint over the values of these attribute, namely

adjunct and full-time, has to be defined in order to satisfy the given policy. Numerous studies have researched how to specify and enforce such constraints in ABAC systems [2, 3, 7, 11]. To the best of our knowledge, the proposal of Bijon et al. is the most comprehensive work to date on constraint specifications in ABAC [2, 3]. Therefore, we adopted their definition of constraints in our work.

Bijon et al. presented a policy specification language called attribute-based constraint specification language (ABCL). The key idea of ABCL is to define constraints using the conflicting relations among attribute values which, in turn, can be used to express notions such as mutual exclusion, preconditions, and obligations amongst attribute values. The authors have identified two dimensions or factors to define four levels of conflict: the number of attributes and the number of entities involved in the conflict relation. Table 1 defines the four levels of conflict, referred to as level 0 to 3, and provide an illustrative example for each level. The primary components of ABCL's notion of constraints are the relation-sets and the ABCL expressions [2, 3]. The relation-sets contain the conflicting values and the expressions are used to precisely specify constraints based on these conflicts as demonstrated in the following examples.[3]

S1: A faculty member cannot hold tenure and non-tenure track positions simultaneously.
RS: $R1 = \{$'tenure', 'non-tenure'$\}$
C1: $|R1 \cap \text{track}(\mathbf{OE(U)})| \leq 1$
S2: An adjunct faculty member should not be considered a full-time employee.
RS: $R2 = \{$'adjunct' , 'full-time'$\}$
C2: $|R2 \cap \text{position}(\mathbf{OE(U)}) \cup \text{employmentType}(\mathbf{OE(U)})| \leq 1$
S3: Only one faculty member can hold the chair position.
RS: $R3 = \{$'chair'$\}$
C3: $|\mathbf{assignedEntities}_{U,position}(\text{'chair'})| \leq 1$
S4: Only one full-time faculty member can hold the chair position.
RS: $R4 = \{$'full-time' , 'chair'$\}$
$\text{U}_{\text{chair}} = \mathbf{assignedEntities}_{U,position}(\text{'chair'})$
C4: $|\mathbf{assignedEntities}_{U_{chair},employmentType}(\text{'full-time'})| \leq 1$

The goal of this work is to automatically identify the elements of the relation-set, denoted as R, along with the limitation, if any, on the number of entities subject to the constraint, as discussed in Section 3.

2.2 Natural Language Parsing Strategies

This subsection presents two natural language parsing strategies that are essential for NLACPs analysis.

Dependency parsing: is a way of representing the syntax of a sentence in a tree-like structure [14]. A dependency parse tree is a directed graph in which

[3] In these examples *track*, *position* and *employmentType* are attribute names. According to the constraints specification language, ABCL [2, 3], attributeName(**OE(U)** is an expression that computes all possible values of the respective attribute while **assignedEntities** returns the entities that satisfy certain attribute values.

Table 1. The four levels of conflict that form ABAC constraints as proposed by [2,3]

Conflict	Definition	Example
Level 0	The conflicts occur among values of a **single** attribute and the constraints apply on each **entity individually**	A faculty member cannot hold **tenure** and **non-tenure** track positions simultaneously.
Level 1	The conflicts occur among values of **multiple** attributes and the constraints apply over each **individual entity**	An **adjunct** faculty member should not be considered a **full-time** employee.
Level 2	The conflicts occur among values of a **single** attribute and the constraints apply on **multiple entities**	Only one faculty member can hold the **chair** position
Level 3	The conflicts occur among values of a **multiple** attribute and constraints apply on **multiple entities**	Only one **full-time** faculty member can hold the **chair** position.

each node represents a word, whilst edges represent the grammatical relationship between two words.

Semantic role labeling (SRL): also known as semantic parsing, decomposes a sentence into its semantic constituents (arguments). The decomposition occurs with respect to each target element (predicate) in the sentence. Then, the semantic role of each argument is recognized with regard to the same target. In other words, the SRL captures who did what to whom, how, when and where for each predicate regardless of the syntactic structure of the sentence [12].

2.3 Sequential Labeling

In natural language processing, a sequence labeling task consists of assigning labels to a sequence of words [24]. In our case, the sequence of words is the NLACP sentence, and the labels indicate words belonging to constraints. For example, the word sequence (w_1, w_2, w_{i},. . . , w_{n}) of a policy sentence is labeled with (l_1, l_2, l_{i}, . . . , l_{n}), where l_{i} is the label assigned to the word w_{i}. Each label is selected from a tag set, typically {B, I, O}, to denote the Beginning, the Intermediate and the non-constraint (Outside) tokens.[4]

Recurrent Neural Networks (RNNs) have shown great promise in this task; we particularly used the Bidirectional LSTM (BiLSTM) model as it has been reported to achieve state-of-the-art performance [9]. More explanation on BiLSTM is given in Section 3.

3 The Proposed Methodology

On the basis of the work of Bijon et al. [2, 3], ABAC constraints are defined using the notion of conflict. Accordingly, to extract constraints from NLACPs one should first extract the conflicting factors that form the constraints (see examples in Subsection 2.1). Hence, the question becomes how to pinpoint the conflicts in NLACPs. Since different combinations of the conflict factors lead to

[4] A token is usually defined as "an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing."

Table 2. Examples of annotated policy sentences using our annotation scheme. Bold type denotes the labeled segment of each constraint

Annotation Example	Explanation
A faculty member **cannot hold tenure and non-tenure track positions** simultaneously.	The constraint is composed of two conflicting values of a single attribute, name it *track*. Hence, it is a level 0 conflict. The annotated span contains the two conflicting values which are "tenure" and "non-tenure."
An adjunct faculty member **should not be considered a full-time employee.**	The constraint is composed of two conflicting values across two attributes, name them *employmentType* and *position*. Hence, it is level 1 conflict. The annotated span contains the portion of conflict that is under constraint. The conflicting values are "full-time" and "adjunct."
Only one faculty member can hold the chair position.	The constraint is composed of a single attribute, name it *position*, with the value of "chair". And it applies on multiple entities, i.e., "faculty members." Hence, it is a level 2 conflict. The annotated span is the portion of constraints that expresses the restriction over the number of entities.
Only one full-time faculty member can hold the chair position.	The conflict occurs between multiple attributes, name them *employmentType* and *position*, and applies on multiple entities. Hence, it is a level 3 of conflict. The annotated span is the portion of constraints that expresses the restriction over the number of entities. The conflicting values are "full-time" and "chair".

different realizations of constraints, we define our annotation scheme based on the levels of conflict discussed in [2, 3]. Then, we design our constraints extraction framework according to the annotation scheme used in the training data, i.e. policy sentences. Particularly, we divide the extraction task into two main phases: identification and normalization. In the former phase, the effort is focused on identifying the right boundary of conflicting values within a policy sentence. In the normalization phase, the focus is on extracting the actual conflicting values that pose a constraint, i.e., the elements of the relation-set, see Subsection 2.1. In the rest of this section, we provide detailed description of constraints annotation scheme, constraints identification phase and constraints normalization phase.

3.1 Annotation Scheme

The question here is how should we annotate the NLACPs to capture different levels of conflict that form ABAC constraints. Since it is extremely challenging to come up with a one-fits-all annotation rule that captures different variations of constraint expressions, we define our annotation scheme based on the level of conflict. Particularly, we identify the level of the conflict, ranging from 0 to 3, for each sentence as it would be done manually by a security architect. Then, we select the portion of policy text that informs our decision regarding the conflict level. We further explain our annotation scheme by examples in Table 2. In each example, we encode the presence of constraints by marking the span of text that best expresses the one factor that is likely to indicate the level of conflict, i.e. values that are subject to constraints for levels 0 and 1 or the restriction on the number of entities considering levels 2 and 3.

To codify the presence of constraints in NLACP, each sentence is represented using the BIO format (Beginning, Inside, Outside). A token is labeled with B-

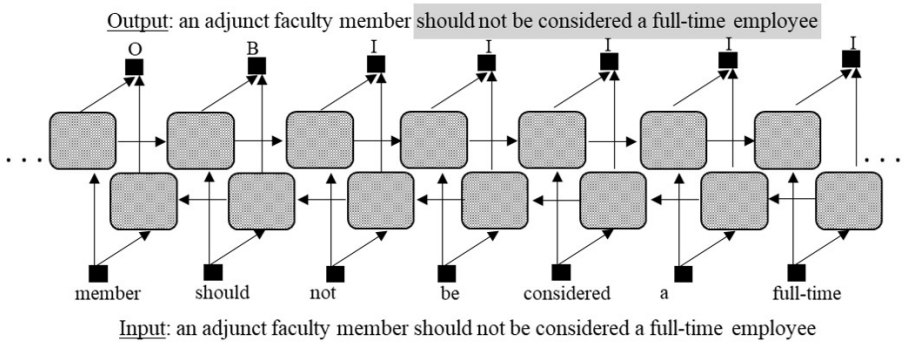

Fig. 1. BiLSTM when applied in the constraints identification task; tokens belonging to the constraint are tagged with the labels B and I. The shaded span of text in the input-output example indicates the output of the BiLSTM model which defines the starting point of the normalization phase as shown in Figures 2 and 3

label if the token is the beginning of a constraint, I-label if it is inside a constraint but not the first token within the constraint, and O-label otherwise.

3.2 Constraints Identification

This phase focuses on the automated detection of constraint expressions within the policy sentence. Hence, the effort is concentrated on predicting their boundaries in the text. In this sense, the constraints identification step can be regarded as a sequence labeling task leading to the choice of BiLSTM, amongst the RNN family models, as it yields the best results. Figure 1 visualizes the abstract functioning of a BiLSTM model when employed in the constraints identification task. The core of our sequence labeling model consists of:

- **Input layer:** This layer forms the real-valued vector representation of individual words in the NLACP sentence using word embeddings. The embeddings can either be learned as a part of the model or loaded from a pretrained word embeddings. Following the recommendation of Chen et al. [4], which calls for the latter approach, we build our model using GloVe pretrained word embeddings. The output of this layer is a matrix $\mathbf{x} = [x_1, x_2, x_t, \ldots, x_n]$. The size of $\mathbf{x}$ is $m \times n$ where m is the dimensionality of the word embedding vectors, and n is the length of the input sequence $\mathbf{x}$.
- **Bidirectional LSTM (BiLSTM) layer:** The BiLSTM layer is an extension of the traditional LSTM layer that further improves the performance on sequential learning problems. The core idea of BiLSTM is that it trains two, instead of one, LSTMs; one learns the sequential information of an input as-is, while the other learns information encoded in the reversed form of the sequence. That is, when a classical LSTM is fed with an ACP sentence, the LSTM computes a representation of the left context of the sentence at each word, denoted as

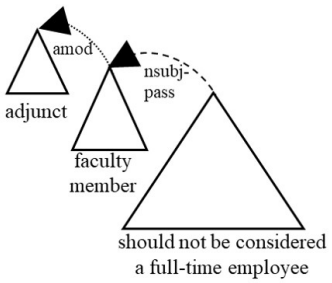

Fig. 2. Collapsed dependency tree representation of "an adjunct faculty member should not be considered a full-time employee."The relation-set R containing the conflicting values is R={adjunct,full-time}

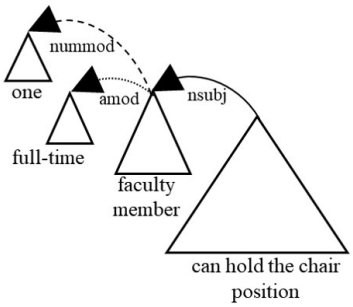

Fig. 3. Collapsed dependency tree representation of "One faculty member can hold the chair position."R={full-time,chair} second element here is captured using SRL tagger

.

$\overrightarrow{h}_t$. Naturally, generating a representation $\overleftarrow{h}_t$ to capture right context as well using a backward LSTM is likely to add useful information. The combination of both representations results in a vector $h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$. This representations effectively encodes the surrounding context of a word, which is useful for numerous sequential learning tasks.

– **Output layer:** In this step, the hidden state vector $\mathbf{h}=[h_1,h_2,h_t, ...,h_n]$ that is obtained from the BiLSTM layer is fed into a fully connected layer to predict or assign a label for each word indicating whether it is inside, a beginning or outside of a constraint expression as shown in Figure 1.

3.3 Constraints Normalization

The input to this phase is a policy sentence in which the sequence of words associated with constraints is identified as discussed in Subsection 3.2.

The goal is to extract the exact conflicting values as well as the restrictions, if any, over the number of entities that are subject to the constraint. In this work, we address this task using a heuristic-based approach. The heuristics have been designed using a combination of the findings of our earlier work in extracting ABAC attributes from NLACPs [1] and the observations gained from our manual analysis of constrained NLACPs. The proposed heuristics are applied as follows.

First, we analyze the dependency tree (DT) representation of the sentence to determine the level of conflict. If none of the words belonging to the segment identified in the previous phase is marked as a numerical modifier, i.e. nummod, or indicates an exact quantity, e.g. the words single, twice etc., then the conflict is regarded as either level 0 or level 1. Otherwise, the conflict is considered as either level 2 or 3. In the former case, i.e., level 0 or 1, the labeled segment

consists of one or more conflicting values as specified in the annotation scheme (see the first two examples in Table 2). To locate the actual values within the segment, we split items combined with commas, "and", "or" or other delimiters. For each split, we search for noun phrases[5] and named entities (NEs)[6]; and, we filter out stop words.

Next, we trace the labeled segment, e.g., "should not be considered a full-time employee," to the corresponding policy element, e.g., "faculty member", that it modifies. The tracing here is done using the DT representation of the sentence as shown in Figure 2 by the dashed arrow. After identifying the policy element, the goal is to extract its attributes, if any. These attributes are to be added to the set of conflicting values, denoted as R. To address this task, we use the list of patterns, identified in our earlier study of ABAC attributes extraction [1], as patterns encoding subject-attribute and object-attribute relations. The result of this step is illustrated by the dotted arrow in Figure 2.

For conflicts of levels 2 or 3, the labeled segment contains the restriction over the number of entities involved in the policy as shown in the last two examples in Table 2. Hence, in this case, the analysis begins with the word marked as the numerical modifier in the DT representation. This word is traced back to the element it modifies as shown in Figure 3 with the dashed arrow between "one" and "faculty member". With this, we automatically infer restrictions over the number of entities, e.g., (1, faculty member). To extract the attributes, if any, of the policy element under restriction we again use the same set of subjects and objects attributes patterns as shown with the dotted arrow in Figure 3. To capture other conflicting attribute values, we apply the SRL tagger on the policy sentence. The resulting arguments— i.e., the who, what, how, when and where— that co-occur with the element under restriction are then directly mapped to these values. The intuition is that a conjunction of these arguments needs to be satisfied to trigger the constraint. Consider for instance the policy sentence, "Only one full-time faculty member can hold the chair position". The SRL tagger captures "Only one full-time faculty member" and "the chair position" as the who and what arguments of the predicate "hold". Since "the chair position" co-occurs with the detected portion of the constraint expression, i.e., "Only one full-time faculty member", as arguments to the same predicate, it is considered as its conflicting value; and, so should be added to the relation-set R.

4 Experimental Results and Performance Evaluation

In this section, we present the evaluation conducted to assess the effectiveness of our proposed approach in addressing the constraints identification and normalization tasks. We begin with a description of the our dataset followed by an

[5] A noun phrase is a sequence of words consisting of a head noun and zero or more modifying adjectives and/or nouns.

[6] A named entity is a real-world object, such as persons, locations, organizations, products, etc.

overview of the general evaluation criteria applied in this study and the experiments and the experimental results.

4.1 Dataset

To evaluate our proposed approach, we constructed a dataset from real-world policy documents from educational institutions, specifically colleges and universities. These documents provide among other things a detailed description of authorizations rules for a wide variety of departments, including Human Resources, Information Technology, Risk Management Services, Faculty and Student Affairs, Administration, Intellectual Property, Technology Transfer, and Equity Development. The documents are written in English. Since not all the sentences in these document express constraints, we manually read through them and collected those that do. The collection process resulted in 801 occurrences of constraints in 747 natural language access control policy (NLACP) sentences. Each sentence contains at least one constraint. Constraints were annotated as discussed in Subsection 3.1. All resulting constraints expressions are multi-token, of which 687 expressions span 5 or more tokens and some span as many as 15 tokens.

4.2 Evaluation Criteria

To evaluate the performance of the constraints identification phase, we calculate the precision, recall, and F1 score over the tokens correctly classified as part of the constraints and over the predicted constraints. The token-level evaluation enables a more thorough understanding of the per-class performance behavior. It answers questions along the line of: How does the model perform at predicting the beginning and intermediate tokens of a sequence? Or is the model biased towards the majority of the non-constraint tokens? The latter, on the other hand, reflects the predictive power of the model at the constraint-level. For this purpose, we use two different definitions of matching: strict and sloppy matching [6]. With strict matching, a predicted constraint is considered correct if and only if its boundary (i.e., the beginning and end) exactly matches the correct constraint. With sloppy matching, a predicted constraint is considered correct if it overlaps with the correct constraint.

To evaluate the performance of the normalization phase, we also use the notion of precision, recall and F1 score. The evaluation measures are computed based on the count of true positives, false positives and false negatives. These counts are determined by comparing the manually identified conflicting values, posing constraint, in each policy sentence against the automatically extracted ones.

4.3 Experiment and Experimental Results

Next, we present the experiments that we conducted to assess the effectiveness of our constraints extraction framework. Through the experiments, we address the following research questions:

- **RQ1:** How effectively, in terms of precision, recall and F1-score, does the constraint identification phase perform in capturing constraints boundary within NLACP sentences ?
- **RQ2:** How effectively, in terms of precision, recall and F1-score, does the constraint normalization phase perform in extracting the conflicting values that form constraints from NLACP sentences?

To answer RQ1, we split our dataset into three separate sets of training, validation, and testing with the proportion of (70:10:20). Training and validation sets were used to adjust the BiLSTM parameters and to decide on the word embedding method, vector length and embedding's source that better fit our data. The testing set, on the other hand, was used to evaluate the performance of the final model. To configure the model, we set one hyper-parameter value at a time. Our default settings: dropout = 0, decay rate = 0, number of BiLSTM cells (i.e., layers) = 1, and GloVe (Common crawl) with 300 dimensions. To determine which word embeddings better capture the semantics of words in our data, we conducted several experiments using pretrained word vectors provided by Mikolov et al. [16] and Pennington et al. [20]. Mikolov's vector representations were trained on Google News using Word2Vec approach whereas Pennington's embeddings were trained on Common Crawl and Wikipedia using GloVe method. Our model performs the best on the validation set when using the word embeddings trained on Wikipedia with 300 dimensions; thus, we use this setting in our final model. Similarly, we ran several experiments to adjust the number of BiLSTM cells and the size of each one, droupout, decay rate and activation function and ultimately set these values as 3, 100 units, 0.1, 0.00001 and tanh, respectively. The detailed results on the validation set were omitted due to space limit.

Once the BiLSTM parameters were configured, we tested the resultant model on our test set. To further understand the difficulty of the problem and to better assess the predictive power of our model, we established a comparative baseline using a conditional random field (CRF) sequence labeler [15]. Note that the CRF sets a strong baseline as it has been successfully applied to sequence labeling problem in many fields including natural language processing [13]. The L1 and L2 regularization parameters of the CRF were set to 0.0001 and 0.0, respectively, following the same tuning procedure applied on the BiLSTM.

We compared the results obtained with the BiLSTM against the results obtained with the CRF which was trained on the same representation of word embeddings. As mentioned in Subsection 4.2, the evaluation measures were defined over two levels of prediction, namely sequence and token levels. Table 3 shows how our model performs against the baseline at predicting constraint sequences. The sloppy and strict sequence matching conditions provide an approximate upper and lower bound of the performance. In order to analyze the performance between these two extreme conditions of matching, we introduce a matching threshold. This threshold determines a minimum size of overlapping that a sloppy matching condition needs to satisfy for accurate predictions.

Table 3. Results of the sequence-level predictions by the means of precision, recall, and F1-score when using our model (the BiLSTM-based) and the baseline (CRF)

Matching \ Model		BiLSTM			CRF		
Type	Threshold	P	R	F1	P	R	F1
Strict	N/A	0.58	0.56	0.57	0.30	0.29	0.30
Sloppy	unconstrained	0.97	0.93	0.95	0.96	0.92	0.94
	25%	0.95	0.92	0.93	0.91	0.87	0.89
	50%	0.94	0.91	0.93	0.86	0.82	0.84
	75%	0.93	0.89	0.91	0.83	0.80	0.81

Table 4. Results of the token-level predictions by the means of precision, recall, and F1-score when using of our model (the BiLSTM-based) and the baseline (CRF)

Model \ Class	BiLSTM			CRF		
	P	R	F1	P	R	F1
B-label	0.83	0.80	0.82	0.72	0.69	0.71
I-label	0.83	0.90	0.86	0.79	0.77	0.78
O-label	0.98	0.97	0.98	0.90	0.91	0.91

Several observations can be made from Table 3. First, concerning the strict matching criterion, our model significantly outperforms the baseline (0.57 vs. 0.30 F1-score). On the other hand, with the unconstrained sloppy matching, both models have shown comparable performance (0.95 and 0.94 F1-score). However, the results observed under the sloppy matching condition differ once we imposed constraints on the overlapping sequences. Two general observations are noted regarding the constrained sloppy matching; (1) The overall performance is proportional to the inverse of the overlapping threshold, i.e. the higher the threshold the lower the performance of both models; (2) The drop observed with our model is not as steep as with the baseline. Using our model, the drop in F1-score is about 4% as we change the settings from constrained to unconstrained matching with a matching threshold of 75%. However, a similar change causes the F1-score of the baseline to drop by about 13%. To justify these observations, we refer to the token-level predictions (i.e., Beginning, Inside, and Outside labels) as shown in Table 4. Considering, the relatively high performance shown by our model at the token-level predictions as well as at sequence predictions under the constrained sloppy matching condition, one can assume that in the vast majority of cases our model is capable to capture at least 75% of the span of text marked as a constraint. Hence, the relatively low scores shown in the case of the strict matching can be attributed to erroneous predictions of at most 25% of the constraint. In other words, our BiLSTM successfully detects most tokens ($\geq$ 75%) that are constraints with an F1 score of 0.91.

To answer RQ2, we employed the set of rules defined in the normalization phase, see Subsection 3.3. For implementation, we used spaCy[7] for dependency

[7] https://spacy.io/

parsing and practNLPTools[8] for semantic parsing. The rules achieved a precision of 0.93, recall of 0.89 and F1 score of 0.90 in extracting the exact conflicting values, forming constraints, from each policy sentence. Overall, the results indicate the soundness of the proposed rules. A closer inspection of the incorrect extractions shows that in the majority of cases erroneous results were due to errors in parsing. Enhancing the parsing effectiveness is most likely to improve the performance [5].

5 Related work

Several studies have been focused on specifying constraints in an ABAC system. Jin et al. have proposed ABACα, a policy specification language, that can also specify constraints on attributes assignment [11]. In ABACα, the set of constraints are defined as the set of values a subject may have given that the subject has already been assigned a particular set of attribute values. Such definition of constraint is regarded as event dependent. Unlike ABACα, Attribute-based Constraint Specification Language (ABCL), presented by Bijon et al., can be used to represent different kinds of conflicting relations among attributes in a system [2,3]. The constraints in ABCL are event independent and are to be uniformly enforced regardless the event causing an attribute value to change. The enforcement of ABCL constraints and its performance is discussed in [2]. Jha et al. have introduced the problem of separation of duty (SoD) specification, verification, and enforcement in ABAC systems [10]. Helil et al. have defined ABAC constraints based on the potential relationships within and across the sets of subjects and objects [7]. Despite the importance of developing formal constraints specification languages, the manual effort involved in the actual process of defining constraints can be difficult, labor-intensive, and error-prone. Our work takes a step to facilitate this process, i.e., converting natural language constraints to machine-readable form.

The necessity to automate the manual analysis of natural language requirement documents has motivated several studies in developing automated tools to derive elements of access control rules directly from the raw text [1, 18, 19, 22, 23, 25]. Previous research efforts have mostly been restricted to three areas: (1) automatic identification of the access control policy sentences from natural language artifacts, (2) automatic extraction of the —subject, resource and action— triples from these sentences, and (3) automatic extraction of the attributes of these elements. Our current work goes beyond the previous efforts in this area by addressing the automated extraction of constraints from natural language policies. This contribution is essential to serve the ultimate goal of developing a comprehensive top-down ABAC policy engineering ecosystem.

[8] https://github.com/biplab-iitb/practNLPTools

6 Conclusion and Future Work

In this work, we designed a two-phase framework to extract ABAC constraints from unstructured NLACPs. We modeled the first phase of the framework, i.e., constraints identification, as sequential labeling problem leading to the choice of BiLSTM. We addressed the second phase, i.e., constraints normalization, using heuristics. The trained BiLSTM successfully detects at least 75% of tokens that are constraints in each policy sentence with F1 score of 0.91. The proposed normalization heuristics, on the other hand, achieved F1 score of 0.90 in extracting the conflicting values from each NLACP sentence. We also designed an annotation scheme that captures constraints expressions within NLACPs. This is an essential contribution to create an annotated corpus that allows natural language constraint analysis studies. The output of the current model is a set of conflicting values that form a constraint, i.e. the relation set R. As a future work, we will study the possibility to incorporate techniques proposed in [17] to integrate a formal analysis component -with natural language front-end- to the proposed framework in order to generate a formal representation of the constraints using the relation set R.

References

1. Alohaly, M., Takabi, H., Blanco, E.: A deep learning approach for extracting attributes of abac policies. In: Proceedings of the 23rd ACM Symposium on Access Control Models and Technologies (2018)
2. Bijon, K.Z., Krishman, R., Sandhu, R.: Constraints specication in attribute based access control. Science **2**(3), pp–131 (2013)
3. Bijon, K.Z., Krishnan, R., Sandhu, R.: Towards an attribute based constraints specification language. In: Social Computing (SocialCom), 2013 International Conference on. pp. 108–113. IEEE (2013)
4. Chen, D., Manning, C.: A fast and accurate dependency parser using neural networks. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 740–750 (2014)
5. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: Proceedings of the conference on empirical methods in natural language processing. pp. 1535–1545. Association for Computational Linguistics (2011)
6. Franzén, K., Eriksson, G., Olsson, F., Asker, L., Lidén, P., Cöster, J.: Protein names and how to find them. International journal of medical informatics (2002)
7. Helil, N., Rahman, K.: Attribute based access control constraint based on subject similarity. In: Advanced Research and Technology in Industry Applications (WARTIA), 2014 IEEE Workshop on. pp. 226–229. IEEE (2014)
8. Hu, V.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., et al.: Guide to attribute based access control (abac) definition and considerations (draft). NIST special publication **800**(162) (2013)
9. Huang, Z., Xu, W., Yu, K.: Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)

10. Jha, S., Sural, S., Atluri, V., Vaidya, J.: Specification and verification of separation of duty constraints in attribute-based access control. IEEE Transactions on Information Forensics and Security **13**(4), 897–911 (2018)
11. Jin, X., Krishnan, R., Sandhu, R.: A unified attribute-based access control model covering dac, mac and rbac. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 41–55. Springer (2012)
12. Johansson, R., Nugues, P.: Dependency-based semantic role labeling of propbank. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 69–78. EMNLP '08, Association for Computational Linguistics, Stroudsburg, PA, USA (2008), http://dl.acm.org/citation.cfm?id=1613715.1613726
13. Kang, T., Zhang, S., Xu, N., Wen, D., Zhang, X., Lei, J.: Detecting negation and scope in chinese clinical notes using character and word embedding. Computer methods and programs in biomedicine **140**, 53–59 (2017)
14. Kübler, S., McDonald, R., Nivre, J.: Dependency parsing. Morgan & Claypool Publishers (2009)
15. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001)
16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
17. Miyao, Y., Butler, A., Yoshimoto, K., Tsujii, J.: A modular architecture for the wide-coverage translation of natural language texts into predicate logic formulas. In: Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation (2010)
18. Narouei, M., Khanpour, H., Takabi, H., Parde, N., Nielsen, R.: Towards a top-down policy engineering framework for attribute-based access control. In: Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies. pp. 103–114. ACM (2017)
19. Narouei, M., Takabi, H.: Towards an automatic top-down role engineering approach using natural language processing techniques. In: Proceedings of the 20th ACM Symposium on Access Control Models and Technologies. pp. 157–160. ACM (2015)
20. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
21. Singh, M.P.: Ahcsabac: Attribute value hierarchies and constraints specification in attribute-based access control. In: Privacy, Security and Trust (PST), 2016 14th Annual Conference on. pp. 35–41. IEEE (2016)
22. Slankas, J., Williams, L.: Access control policy identification and extraction from project documentation. Academy of Science and Engineering Science (2013)
23. Slankas, J., Xiao, X., Williams, L., Xie, T.: Relation extraction for inferring access control rules from natural language artifacts. In: Proceedings of the 30th Annual Computer Security Applications Conference. pp. 366–375. ACM (2014)
24. Tjong Kim Sang, E.F., De Meulder, F.: Introduction to the conll-2003 shared task: Language-independent named entity recognition. In: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4. pp. 142–147. Association for Computational Linguistics (2003)
25. Xiao, X., Paradkar, A., Thummalapenta, S., Xie, T.: Automated extraction of security policies from natural-language software documents. In: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. p. 12. ACM (2012)