Identification of Access Control Policy Sentences from Natural Language Policy Documents

Masoud Narouei, Hamed Khanpour, Hassan Takabi

Identification of Access Control Policy Sentences from Natural Language Policy Documents

Masoud Narouei, Hamed Khanpour, and Hassan Takabi

Department of Computer Science and Engineering,
University of North Texas,
Denton, TX, USA,
{Masoudnarouei,Hamedkhanpour}@my.unt.edu, Hassan.Takabi@unt.edu

Abstract. Access control mechanisms are a necessary and crucial design element to any application's security. There are a plethora of accepted access control models in the information security realm. However, attribute-based access control (ABAC) has been proposed as a general model that could overcome the limitations of the dominant access control models (i.e, role-based access control) while unifying their advantages. One issue with migrating to an ABAC model is the information that needs to be encoded in the model is typically buried within existing natural language artifacts, hence difficult to interpret. This requires processing natural language documents and extracting policies from those documents. Software requirements and policy documents are the main sources of declaring organizational policies, but they are often huge and consist of a lot of general descriptive sentences that lack any access control content. Manually processing these documents to extract policies and then using them to build a model is a laborious and expensive process. This paper is the first step towards a new policy engineering approach for ABAC by processing policy documents and identifying access control contents. We take advantage of multiple natural language processing techniques including pointwise mutual information to identify access control policy sentences within natural language documents. We evaluate our approach on documents from different domains including conference management, education, and healthcare. Our methodology effectively identifies policy sentences with an average recall and precision of 90% on all datasets, which bested the state-of-the-art by 5%.

Keywords: Access control policy . Attribute-based access control . Policy engineering . Natural language processing

1 Introduction

A fundamental management responsibility is securing information and information systems. Almost all of the applications that deal with safety, privacy, defense and even finance include some form of access control, which regulates who or what can view or use resources in a computing environment. Access control policies (ACP) are those rules that a corporation exerts in order to control access

to its information assets. They determine which activities are allowed for authorized users, controlling every attempt by a user for accessing system resources. They also reveal a lot of information about the internal processes within an organization, which sometimes mirror the structure of the organization. The risks of not using adequate access control policies range from inconvenience to critical loss or corruption of data. A big challenge for security administrators while implementing an access control model is to properly define ACPs, especially in large corporations. Most of these corporations have high-level requirement specification documents that specify how information access is manipulated and who, under which circumstances, can access what asset. All US federal agencies are required to provide information security by the "Federal Information Security Act of 2002" [2], and policy documentation is part of that requirement. Although it is not necessary for private industry to prepare such documentation, the remarkable costs affiliated with recent cyber-attacks have forced many corporations to record their security policies. In addition, recording security policies aid corporations in transitioning from access control lists into a more robust access control model such as Attribute-based Access Control (ABAC) with more ease. In this paper, we refer to these documents as natural language access control policies (NLACPs), which are described as "statements governing management and access of enterprise objects. They are human expressions that can be translated to machine-enforceable access control policies" [10].

An issue with NLACPs is they are usually declared in human understandable terms, are unstructured, and also may be ambiguous, hence it is difficult to encode them directly in a machine-enforceable form. In our previous work [19], we addressed this issue by proposing semantic role labeling technique, where we were able to process sentences and extract necessary elements such as subject, object and action to create machine-enforceable ACPs. However, prior to identifying ACP elements, one has to first process the unstructured NLACP documents in order to identify those sentences that express policies. NLACPs are often huge and consist of many sentences. Several of these sentences are general descriptive sentences that lack any access control policy content (non-ACP sentences). The process of manually sifting through NLACPs to extract the buried ACPs is very laborious and error-prone.

This paper is the first step towards our ultimate goal of building an ABAC system from existing information. Current ABAC solutions try to convert already existing policies in the form of ACL [32], RBAC, etc. to an equivalent ABAC model. However, the previous work ignores an important source of information, NLACP documents, in the process of building an ABAC model. In this paper, we aim to propose a new technique to solve this issue in order to make it easier for corporations to extract ACPs from NLACPs. We will limit our discussion to identifying ACP sentences and separating them from other non-informative, irrelevant sentences. In the future, we aim to use the extracted ACP sentences to build a new ABAC model. Our proposed technique takes advantage of four different types of features in order to come up with a final discriminating feature set. These features include pointwise mutual information (PMI) features,

security features, syntactic complexity features and dependency features. To the best of our knowledge, this is the first report that elaborates on the effectiveness of using four different feature vectors to extract policies.

The contributions of this paper are as follow:

- We take the first step towards proposing a new policy engineering approach for ABAC by processing NLACP documents and extracting policy contents.
- We introduce a new technique to effectively distinguish ACP sentences from non-ACP sentences.
- We introduce four different types of features in order to come up with a final discriminating feature set.
- We introduce the first publicly available policy dataset to encourage more research on this area.

The rest of this paper is organized as follows: We begin with a discussion of related work in section 2. Section 3 will discuss some background concepts and then section 4 will present our methodology. The experiments are presented in section 5, and finally, conclusion and future works wraps up the paper.

2 Related Work

Previous work in the literature took advantage of predefined patterns and combining machine learning approaches to find access control policy sentences. Xiao *et al.* proposed Text2Policy, which employs shallow parsing techniques with finite state transducers to match a sentence into one of four access control patterns [30]. One example of such a pattern is *Modal Verb in Main Verb Group*, which helps recognize that the main verb contains a modal verb, which leads to identifying the sentence as an ACP sentence. If the matching is successful, it uses the annotated portions of the sentence to extract the subject, action, and object from the sentence. Text2Policy does not need a labeled data set. However, it misses ACPs that do not follow one of its four semantic patterns. It is reported that only 34.4% of the identified ACP sentences followed one of Text2Policy's patterns [25].

Slankas *et al.* proposed access control rule extraction (ACRE), a supervised learning approach that uses an ensemble classifier to determine whether a sentence expresses an ACR or not [25]. Their ensemble classifier is composed of a k-nearest neighbors (k-NN) classifier, a naïve bayes classifier, and a support vector machine classifier. To determine how to use these classifiers, they calculated a threshold value for the ratio of the computed distance to the neighbors compared to the number of words in the sentence. If the k-NN classifier's ratio for a sentence is below a threshold value of 0.6, they return the k-NN classifier's predication. Otherwise, they output a majority vote of the three classifiers. Even though this approach performs very well, the k-NN classifier suffers from a slow processing time since it has to compare each sentence to all other sentences to make a decision.

In our previous work, we proposed semantic role labeling (SRL) to automatically extract ACP elements from unrestricted natural language documents, define roles, and build an RBAC model [20]. We did not attempt to identify ACP sentences, but instead used the already extracted sentences by [25] and left implementing the ACP sentence identification step for future work. In this work, we propose our methodology for ACP sentence identification.

The problem of mining ABAC policies from natural language documents has not been studied in the literature. However, there are few works for deriving ABAC policies from request logs. This problem was first investigated in [31]. The authors present an algorithm for mining ABAC policies from logs and attribute data. The algorithm iterates over tuples in the user-permission relation extracted from the log, uses selected tuples as seeds for constructing candidate rules, and attempts to generalize each candidate rule to cover additional tuples in the user-permission relation. Finally, it selects the highest quality candidate rules for inclusion in the generated policy. In [16], the authors proposed a multi-objective evolutionary approach for learning ABAC policies from sets of authorized and denied access requests. They used the same ABAC language and the same case studies of the [31]. They presented a strategy for learning policies by learning single rules, each one focused on a subset of requests, an initialization of the population; a scheme for diversity promotion and for early termination. Most recently, we introduced a policy engineering framework for ABAC where we were able to identify access control policy sentences using deep learning techniques and convert some ACP sentences to ABAC policies using semantic role labeling technique [18]. There are also some other work for inferring RBAC policies from logs for less expressive access control models (e.g., RBAC [7]). Usage of evolutionary techniques for inferring RBAC rules explaining the observed actions in environments with tree-structured role hierarchies was also proposed in [9].

3 Background

This section provides background with regards to PMI, syntactic complexity, ML, and ABAC with the motivation behind using these techniques.

3.1 Pointwise Mutual Information

Pointwise mutual information (PMI) has been extensively applied in information retrieval (IR) and text based applications. PMI was introduced by Turney [27] as an unsupervised learning algorithm for finding synonyms based on statistical information. It gives an estimation of semantic similarity between two words based on word co-occurrence[1] on a very large corpus, e.g., Web, Wikipedia. Given two words t1 and t2, their PMI score is defined as Equation 1:

[1] Frequent occurrence of two words from a text corpus alongside each other in a certain order

$$PMI(t_1, t_2) = \log_2 \frac{P(t_1, t_2)}{P(t_1) * P(t_2)} \quad (1)$$

Where $P(t_1, t_2)$ is the probability of $P(t_1)$ and $P(t_2)$ under the joint distribution and $P(t_1)$,$P(t_2)$ is the probability of each word independently. Thus, if $P(t_1)$ and $P(t_2)$ are independent, $P(t_1, t_2) = P(t_1)P(t_2)$, and PMI is $log(1) = 0$, meaning that $P(t_1)$ and $P(t_2)$ share no information.

We take advantage of PMI scores in order to extract informative features that are relevant to the current task. This will also narrow down the number of features significantly as will be described in section 4.2.2.

3.2 Measures of Syntactic Complexity

ACP sentences usually contain more complex structures such as clauses than non-ACP sentences (e.g., *"If by chance a student is employed at a particular clinic or health care institution for any reason , that student may not be placed at that clinic or institution for any of their clinical practical."* compared to non-ACP content such as *"All Health providers and staff."* Hence, we evaluate syntactic complexity of written English texts to better discriminate ACP sentences from non-ACP contents. Although many different measures have been suggested for characterizing syntactic complexity, previous literature only considered a small set since not many computational tools are available and manual analysis is laborious. Recently Lu [13] proposed a computational system that uses 14 different measures to analyze syntactic complexity in second language writing. These 14 syntactic complexity measures are chosen from a large number of measures that were discussed in [29] and [22].

Table 1. The 14 syntactic complexity measures used in previous research [13]

Mean length of clause	MLC	# of words / # of clauses
Mean length of sentence	MLS	# of words / # of sentences
Mean length of T-unit	MLT	# of words / # of T-units
Sentence complexity ratio	C/S	# of clauses / # of sentences
T-unit complexity ratio	C/T	# of clauses / # of T-units
Complex T-unit ratio	CT/T	# of complex T-units / # of T-units
Dependent clause ratio	DC/C	# of dependent clauses / # of clauses
Dependent clauses per T-unit	DC/T	# of dependent clauses / # of T-units
Coordinate phrases per clause	CP/C	# of coordinate phrases / # of clauses
Coordinate phrases per T-unit	CP/T	# of coordinate phrases / # of T-units
Sentence coordination ratio	T/S	# of T-units / # of sentences
Complex nominals per clause	CN/C	# of complex nominals / # of clauses
Complex nominals per T-unit	CN/T	# of complex nominals / # of T-units
Verb phrases per T-unit	VP/T	# of verb phrases / # of T-units

This system takes input as a plain text file and then counts the frequency of the following nine structures:

- Words (W)
- Sentences (S)
- Verb phrases (VP)
- Clauses (C)
- T-units (T)
- Dependent clauses (DC)
- Complex T-units (CT)
- Coordinate phrases (CP)
- Complex nominals (CN)

Using these calculated frequencies, the system produces 14 indices of syntactic complexity, presented in Table 1. We use these 14 features to measure syntactic complexity of sentences, which will help classification.

3.3 Machine Learning

Machine learning (ML) is a method of data analysis that automates building an analytical model. Using algorithms that iteratively learn from data, ML allows computers to find hidden insights without being explicitly programmed. ML algorithms are often categorized as being supervised or unsupervised. Supervised algorithms can apply what has been learned in the past to new data. Unsupervised algorithms, however, draw inferences from new data. In this work, we have a labeled dataset and hence we use supervised ML algorithms. We will take advantage of two ML algorithms, naïve bayes and support vector machines. The naïve bayes algorithm is based on conditional probabilities. It uses Bayes' Theorem, to find the probability of an event occurring given the probability of another event that has already occurred. Using these probabilities, the algorithm calculates the probability of any sentence being either ACP or non-ACP and then makes the final decision based on the higher probability. We also employ support vector machine, which is a supervised ML algorithm that constructs a hyperplane or set of hyperplanes in a high-dimensional space that is used for classification. The algorithm plots all sentences in a n-dimensional space (where n is number of features) with the value of each feature being the value of a particular coordinate. Then, the algorithm finds the hyper-plane(s) that best separates ACP and non-ACP sentences.

3.4 Attribute-based Access Control

Attribute-based access control (ABAC) is an access control model wherein the access control decisions are made based on a set of attributes, associated with the requester, the environment, and/or the resource itself. An attribute is a property expressed as a name:value pair that can capture identities and access control lists (DAC), security labels, clearances and classifications (MAC), and

roles (RBAC). ABAC was proposed as a general model that could overcome the limitations of the dominant access control models (i.e, discretionary-DAC, mandatory-MAC, and role-based-RBAC) while unifying their advantages. Our central contribution is to take the first step towards proposing a new policy engineering approach for ABAC by processing policy documents and extracting those sentences that exhibit ACP content. We then use these ACP sentences to extract ABAC policies.

4 The Proposed Methodology

In order to effectively distinguish ACP sentences from non-ACP sentences, we take advantage of different NLP and ML techniques. Our proposed system is composed of a set of components that extract various types of features. The overview of the system is presented in Figure 1. In the next sections, we will describe each of these components in detail.

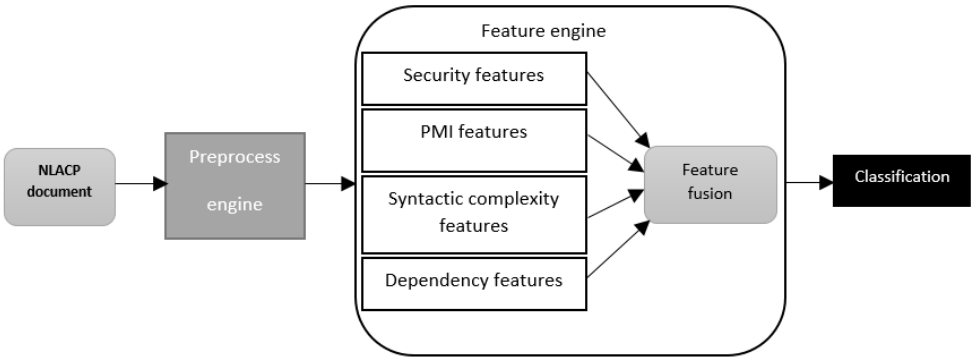


Fig. 1. Overview of the proposed system

4.1 Preprocess Engine

Figure 2 presents part of a large NLACP document [2]. It is obvious that there are many non-relevant contents such as titles, tables, etc that need to be removed. As these formal NLACP documents are usually expressed in PDF format, the first step is to read each PDF document. For this purpose we used Apache PDFBox [1] toolkit, which extracts texts and ignore the other contents such as tables. In

[2] http://policy.unt.edu/sites/default/files/07.022_AdministrativeEntrySearchesUniversityResidenceHalls_2012.pdf

order to parse the extracted text, we fed it into Stanford Corenlp toolkit [15]. The tool split the text by sentence boundaries where each sentence was on a separate line and ended by a period. As many of the extracted sentences are not statements (e.g., titles), we introduce the following equation to filter out everything other than sentences.

$$Ratio(sent) = \frac{Capitals(sent)}{Tokens(sent)}$$

Where *Capitals* stand for the number of capital letters in the sentence (*sent*) and *Tokens* counts for the number of tokens in each sentences. If this ratio is less than 0.6, we consider the sentence for further processing. We used different ratios but 0.6 gave us the most accurate results. We also limited ourselves to sentences with more than 15 characters, which helped us remove more incomplete sentences. After this step, the following four sentences are extracted:

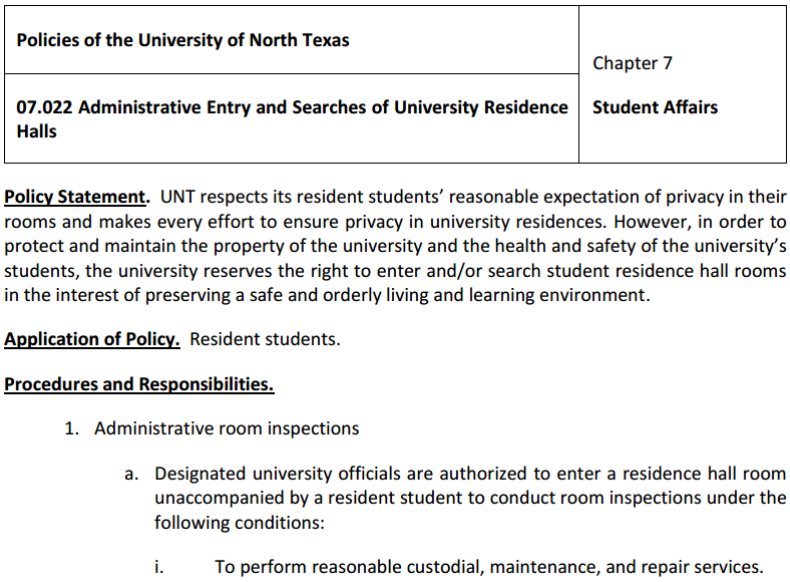
- *The University respects its resident students' reasonable expectation of privacy in their rooms and makes every effort to ensure privacy in university residences.*
- *However, in order to protect and maintain the property of the university and the health and safety of the university's students, the university reserves the right to enter and/or search student residence hall rooms in the interest of preserving a safe and orderly living and learning environment.*
- *Designated university officials are authorized to enter a residence hall room unaccompanied by a resident student to conduct room inspections under the following conditions.*
- *To perform reasonable custodial, maintenance, and repair services.*

Next, the extracted sentences were fed to feature engine in order to extract discriminating features.

4.2 Feature Engine

Feature engine is responsible for extracting various types of features and creating the final feature vector. The following sections will describe each sub-component in detail.

Security features Kong *et al.* proposed a system called AUTOREB that was able to infer the mobile app security behaviors using apps' reviews from different users with an average accuracy of 94% [12]. As their problem is quite similar to ours in terms of classifying security sentences, we decided to take advantage of their proposed methodology to extract more discriminating features. Our adapted methodology is described as follows:
First, we ranked all words in ACP sentences based on their frequencies of occurrence and chose the top 15 most frequent keywords. These 15 keywords were considered as the initial set of our security features. Then, the new keywords



Policies of the University of North Texas	Chapter 7
07.022 Administrative Entry and Searches of University Residence Halls	Student Affairs

Policy Statement. UNT respects its resident students' reasonable expectation of privacy in their rooms and makes every effort to ensure privacy in university residences. However, in order to protect and maintain the property of the university and the health and safety of the university's students, the university reserves the right to enter and/or search student residence hall rooms in the interest of preserving a safe and orderly living and learning environment.

Application of Policy. Resident students.

Procedures and Responsibilities.

1. Administrative room inspections
    a. Designated university officials are authorized to enter a residence hall room unaccompanied by a resident student to conduct room inspections under the following conditions:
        i. To perform reasonable custodial, maintenance, and repair services.

Fig. 2. Part of a NLACP document

were chosen from those that had a high co-occurrence with current keywords in our feature set. If this co-occurrence exceeded a threshold, we added the new keyword into the feature set. To avoid repetitive calculations, at each round we only considered those keywords that were added in the previous round. This process was repeated until no more new keywords were added. In the case of applying the same methodology on any other dataset, the same process can be applied to identify the keywords relevant to that dataset.

PMI features N-grams have been extensively used for text classification as a good syntactic feature. However, they often result in large and sparse feature vectors. By taking advantage of PMI scores, we can limit ourselves to only those words that are informative and correlated to the current task. This will improve the performance of classification algorithm and speed it up considerably. To have an accurate calculation of PMI scores, a large corpus is required. In the original arrangement of PMI by Turney [27], it is required to consult the Web for counting words. However, there are some disadvantages with using the Web. The Web is always changing and the search scheme of commercial search engines is always changing too, which makes it hard to maintain the functionality of the system. For computing semantic relations, it has been shown that Wikipedia is more reliable and effective than a search engine and covers more concepts than WordNet [23]. In order to extract PMI scores, we used two different setups. For the first one, we gathered pre-calculated PMI data through Semilar project

[24]. This data was calculated using whole Wikipedia text (as of Jan 2013). For the second setup, we calculated PMI scores from access control contexts as mentioned briefly in section 3.1 so that it will be more adaptable to our approach. We gathered over 10 MB of text consisting all of our datasets and then calculated PMI scores based on the formula described in section 3.1. To calculate relevant keywords, we adopt the same process that was described in previous section. Starting with the same initial set of 15 keywords, we calculated the PMI score of all words in our dataset. For each setup, we used the corresponding set of PMI scores. If this PMI score was more than a threshold, we added the other word to the feature set too. After the first round, we came up with a new set of features in addition to the initial 15 keywords. Then in the second iteration, we repeated the process using newly found keywords in the previous step. We followed this process until no more features were added to our feature set.

Syntactic complexity features This component computes complexity measures based on the explanations in section 3.2. For each sentence a feature vector of 23 values were created.

Dependency relations Marneffe *et al.* described a methodology for extracting typed dependency parses of English sentences [4]. A dependency parse demonstrates dependencies among single words. It also labels dependencies with grammatical relations, such as subject or indirect object. Consider the sentence: "Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas." The corresponding typed dependency tree is drawn in Figure 3.

Typed dependency parses of ACP sentences capture inherent relations occurring in ACP sentences that can be critical in discriminating them from non-ACP contents. Using this structure, we extracted the ratio of occurrences of each of the following attributes to the length of sentence and used them as features:

- Subject
- object
- auxiliary verb
- verb

Feature fusion This component stacks all the four feature types in a single, long feature vector that were used for classification. For PMI and security features, there are some redundancies in extracted keywords that were removed too.

4.3 Classification

After building the final feature vector for each sentence and creating the final dataset consisting of all vectors, we train a predictive model using ML algorithms. Since our training set has fairly little data, we need to choose a classifier with high bias according to machine learning theory [14]. For this work, we chose

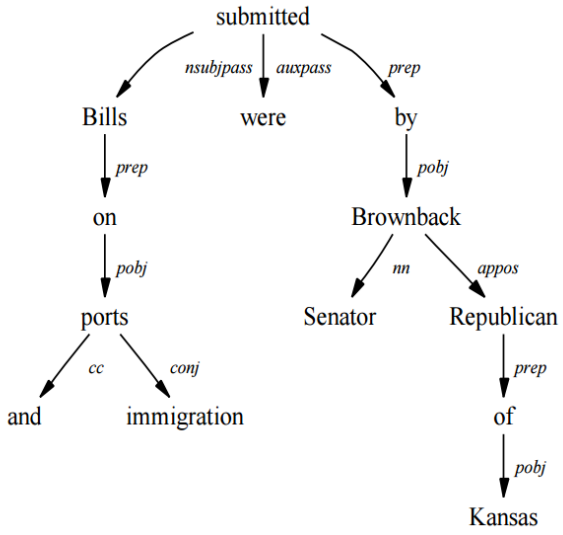


Fig. 3. Typed dependency parse for the sentence "Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas."

naïve bayes as there are many theoretical and empirical results that naïve bayes does well in such circumstances (e.g. [6], [21]). We also employed support vector machines classifier as it has consistently achieved good performance on text categorization tasks (see [11]). The created model will be used to categorize unseen sentences to either ACP or non-ACP.

5 Experiments and Results

5.1 Dataset(s)

The access policy domain suffers from a scarcity of publicly available data for researchers. To help alleviate this issue, we created a dataset to serve the dual purpose of (1) making our evaluation of the proposed method more robust, and (2) providing the research community with more data, which will in turn allow other researchers to evaluate their work both more comprehensively and in direct comparison to others.

We constructed our dataset from real-world policy documents from the authors' home institution. To do this, we gathered over 430 policy documents in PDF format from the university policy office [3], as well as policy documents from the university's health science center [4]. The documents described security access

[3] https://policy.unt.edu/policy-alphabetical/a
[4] https://app.unthsc.edu/policies/Home/ByChapter

authorizations for a wide variety of departments, including humanresources, information technology, risk management services, faculty affairs, administration, intellectual property, technology transfer, and equity development, among others. Altogether, these documents were comprised of more than 21,000 sentences. Since manually labeling the sentences is a labor-intensive process, in this work we limited our data to a randomly selected subset of 1,000 sentences from the full dataset.

The sentences were annotated for the presence of ACP content by two Ph.D. students studying cybersecurity, who are familiar with access control policies and the contexts in which they occur. They were also provided a coding guide that told them what should be considered as an access control policy sentence and what should not. The first author of this paper adjudicated any discrepancies in the annotations after discussing them with both annotators. We computed kappa on the annotations, finding $\kappa = 0.75$ between the two annotators. The final annotated dataset is comprised of 455 ACP sentences and 545 non-ACP sentences. This dataset is available upon request.

5.2 Evaluation Criteria

In order to evaluate results, we use recall, precision, and the F_1 measure. Precision is the fraction of ACP sentences that are relevant, while recall is the fraction of ACP sentences that are retrieved. To compute these values, the classifier's predictions are categorized into four categories. True positives (TP) are correct predictions. True negatives (TN) are sentences that we correctly predicted as not an ACP sentence. False positives (FP) are sentences that were mistakenly identified as an ACP sentence. Finally, false negatives (FN) are ACP sentences that we failed to correctly predict as an ACP sentence. Using these values, precision is calculated using $P = \frac{TP}{TP+FP}$ and recall using $R = \frac{TP}{TP+FN}$. To have an effective model, a high value for both precision and recall is required. Lower recall means the approach could more likely miss ACP sentences while a lower precision implies that the approach could more likely identify non-ACP sentences as ACP sentences. We define F_1 as the harmonic mean of precision and recall, giving an equal weight to both elements. F_1 measure is calculated using the $F_1 = 2 \times \frac{P \times R}{P+R}$ respectively.

Table 2. Obtained results using two classifiers (naïve bayes and support vector machines(SVM)) alongside comparison with baseline and the implemented ensemble classifier

Methodology	Precision(%)	Recall(%)	F_1(%)
ZeroR	30	54	39
Ensemble	74	70	72
SVM	76	76	76
Naïve Bayes	82	80	80

5.3 Experimental Results

After performing the preprocessing, the dataset is divided into 70% training (700 sentences) and 30% testing sets (300 sentences). In order for both datasets to be independent and identically distributed, stratification was performed to make sure the distribution of both classes (ACP and non-ACP) are the same in both train and test sets. Then, the training set is fed to feature engine for feature extraction. The initial set of seeds for all datasets are presented in Table 4. All the experiments were performed using weka toolkit [8], which is a collection of ML algorithms for data mining tasks. We used SMO (support vector machine's implementation in weka) as well as naïve bayes classifier for classification task. We trained both classifiers using the training set. The performance of the system

Table 3. Study document set statistics

Dataset	Domain	# of sentences	# of ACP sentences
iTrust for ACRE	Healthcare	1160	550
iTrust for Text2policy	Healthcare	471	418
IBM Course Management	Education	401	169
CyberChair	Conference Mgmt	303	139
Collected ACP Documents	Multiple	142	114

on the testing dataset is reported in Table 2. The second row shows the baseline results generated using weka's ZeroR classifier. The ZeroR algorithm selects the majority class in the dataset and uses it to make all predictions.

The third row shows the comparison with the ACRE system proposed by Slankas *et al.* [25]. In their ensemble classifier, an object is assigned to the class most common among its k nearest neighbors. The classifier used a modified version of Levenshtein distance as distance metric. The details of this classifier was presented in Section 2. We were not able to receive the source codes for their ensemble classifier and hence we implemented it based on the description provided in their paper as well as the main author's thesis [26]. We did our best to make sure that our implemented ensemble classifier is similar to their methodology. Finally, the fourth and fifth rows present results obtained using our proposed method. As Table 2 shows, our methodology was able to outperform the majority baseline and previous work by a huge margin.

In order to have an exact comparison between our proposed methodology and the ACRE system, we performed experiments on the same datasets that they used in their paper. These datasets were manually labeled by Slankas *et al.* [25] and consisted of five sections, described as follows:

- **iTrust for ACRE**
  iTrust [17] is an open source healthcare application that consists of 40 use cases plus additional non-functional requirements. *iTrust for ACRE* was extracted directly from the project's wiki.

- **iTrust for Text2policy** The second version of iTrust that was taken from the documentations used by Xiao *et al.* [30]
- **IBM Course Management** Eight use cases from the IBM Course Registration System [3].
- **CyberChair** CyberChair documents [28], which has been used by over 475 different conferences and workshops.
- **Collected ACP Documents** A combined document of 142 sentences that were collected by Xiao *et al.* [30].

More information about these dataset(s) can be found in Table 3. The initial seeds for each dataset is presented in Table 4. Using these seeds, each dataset was independently fed to the feature engine and experiments were performed. The obtained results are presented in Table 5. As the table presents, applying our implemented system on all five dataset(s) achieved a macro average of 90% F_1 while the ACRE system achieved 85%.

Table 4. The top 15 frequent keywords for each dataset

UNT Policies	will, must, may, health, student, shall, information, employee, should, science, review, center, policy, university, staff
Collected	can, access, read, if, customer, subject, assign, information, paper, resident, use, patient, medical, may, reps
CyberChair	reviewer, paper, submit, author, assign, number, must, expertise, should, indicate, process, base, overview, topic, abstract
IBM	system, student, course, professor, schedule, registrar, offering, case, use, semester, information, offering, delete , will, select
iTrustforACRE	patient, office, hcp, can, name, date, message, list, appointment, information, user, lab, representative, view, office
iTrustfortext2policy	patient, view, can, message, representative, name, list, system, hcp, date, appointment, data, user, present, administrator

Table 5. Comparison with ACRE system

	ACRE			Proposed system		
Dataset	Precision(%)	Recall(%)	F_1(%)	Precision(%)	Recall(%)	F_1(%)
iTrust for ACRE	90	86	88	89	90	90
iTrust for Text2policy	96	99	98	97	99	98
IBM Course Management	83	92	87	94	93	93
CyberChair	63	64	64	79	74	77
Collected ACP Documents	83	96	89	91	93	92
Average	83	87	85	90	90	90

In order to extract effective PMI scores, we used both pre-calculated PMI values using Wikipedia and also our self-calculated PMI values based on security context. In order to come up with the best threshold, we used ten-fold cross validation on the train set to evaluate the performance using features generated by

each threshold value. Figure 4(a) presents the performance using pre-calculated PMI values from Wikipedia. The best set of features were extracted using a threshold value of 0.4. Figure 4(b) shows the performance using self-calculated PMI values. The best performance (83%) was obtained using a threshold value of 0.8. As these figures indicate, our self-generated PMI scores yield a higher performance, which seems reasonable considering the fact that they were generated from the security context while pre-calculated PMI values were generated using public Wikipedia article, which are general to any method but not specific to our domain. Hence, we used our self-calculated PMI scores with the threshold of 0.8 as the output of PMI component. For security features, the best results were obtained using a threshold value of 15 co-occurrences in the whole context, as Figure 5 shows, hence we used this threshold. It is obvious that the performance decreases considerably after a threshold of 20 co-occurrences since there are not many words that occur together more than 20 times in the dataset.

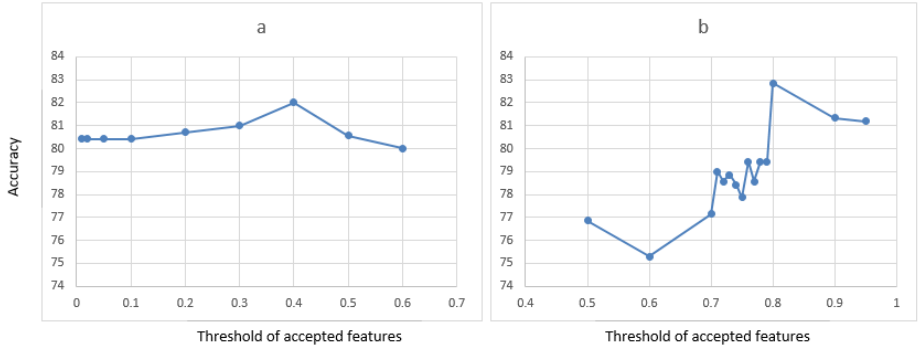


Fig. 4. Threshold analysis using PMI values from Wikipedia (a) and self-calculated PMI values (b)

Table 6. The number of different feature values

Feature type	# of features
Security features	73
Policy PMI	721
Syntactic complexity	23
Dependency features	4

Table 5 shows the final number of features generated by each component while performing the first experiment (policy documents). Overall, 750 features were extracted after combining all of the features together and removing duplicates. These features were used to build a final feature vector for each sentence.

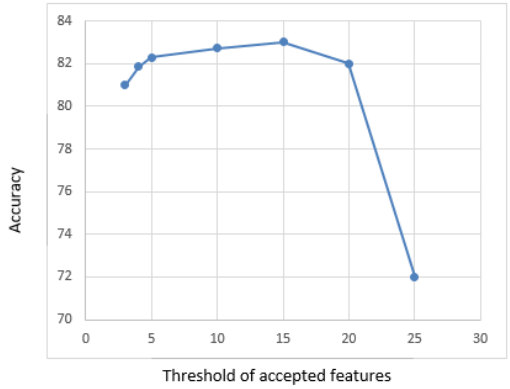


Fig. 5. Threshold analysis of co-occurrences of security features

For all features except complexity and dependency features that had their own calculated values, the presence or absence of the features in the corresponding sentence was considered.

Our dataset consists of ACP and non-ACP sentences both expressed in security context. Even distinguishing an ACP sentence from a non-ACP sentence was sometimes difficult as expressed by our labelers. Our proposed system was able to distinguish between these sentences by 80% F_1 while the state-of-the-art gains around 72% F_1. The strength of our methodology comes from incorporating both lexical and semantic features. As all sentences are expressed in ACP context, using just lexical approach is not sufficient. However, using syntactic complexity features improves the results as ACP sentences usually consist of clauses and more complex structures compared to non-ACP sentences. These structures convey lots of information but are usually ignored using only lexical approaches. The ensemble classifier that was used in the previous work considered the syntax of sentence (using a modified version of Levenshtein distance for comparing words) to identify ACP sentences. This approach performed very well on their reported experiments, but while analyzing policy documents, both syntactic and semantic features proved more fruitful.

As we mentioned in the previous section, we were unable to get their codes, and hence we implemented their methodology in Java language using Stanford Corenlp package. On average, our implementation reported 84% average precision on all five datasets (iTrust for ACRE, iTrust for Text2policy, IBM Course Management, CyberChair, Collected ACP Documents) while their reported average precision was 81%. The F_1, however, was lower as we got around 81% while they reported 84%. The main problem with ensemble classifier was the speed since for each instance, the k-NN classifier needs to compute the distance to all other sentences. This resulted in an average execution time of about 5-6 hours on the five datasets while our proposed methodology runs for less than a

minute given the features, or about 20 minutes for extracting features depending on the size of dataset.

6 Discussion

There are several threats to validity of experiments including lack of representativeness of datasets, identifying a threshold for features and human factors for determining correct ACP sentences from NLACPs. The five datasets used in previous literature covered mostly limited grammars and many of their policies were of similar structure and form, not representing the diversity of policies in real-world. To reduce the threat, we evaluated our approach on policy documents from authors' home institution. These documents covered a large variety of policies ranging from human resources, information technology, risk management services, faculty affairs, administration, intellectual property, technology transfer, and equity development, among others. To further reduce this threat, additional evaluation needs to be done to choose a more representative sample of dataset, instead of choosing sentences randomly. Choosing the proper threshold is another issue as it determines the quality of extracted keywords. To reduce this threat, we examined different threshold values from different ranges. A final threat include human factors for determining correct ACP sentences from NLACPs. To reduce the human factor threats, the sentences were annotated for the presence of ACP content by two Ph.D. students studying cybersecurity, who are familiar with ACPs and the contexts in which they occur. The co-author of this paper adjudicated any discrepancies in the annotations after discussing them with both annotators.

To further evaluate our method's performance on ACP sentences that were previously used in the literature, we performed another experiment. We randomly sampled 250 ACP sentences from the four dataset(s) that were previously described (iTrust, IBM course registration system, cyberchair and collected documents). We also gathered 250 sentences that carry no ACP information from Microsoft research paraphrase corpus [5]. This dataset contains 5,800 pairs of sentences that were extracted from news sources on the web, alongside human annotations indicating whether each pair capture a semantic equivalence relationship. Only one sentence has been extracted from any given news article. Using 10-fold cross validation, our proposed system was able to correctly identify ACP sentences with an accuracy of 94%, which shows applicability of this method in the wild.

7 Conclusion and Future Work

ABAC is a promising alternative to traditional models of access control (i.e., DAC, MAC and RBAC) that is drawing attention in both recent academic literature and industry. However, the cost of developing ABAC policies can be a significant obstacle for organizations to migrate from traditional access control models to ABAC. In this paper, we took the first step towards a new policy

engineering approach for ABAC by processing policy documents and extracting access control contents. We took advantage of multiple natural language processing techniques including pointwise mutual information to identify access control policy sentences. Experimental results yielded an average 90% F_1, which bested the state-of-the-art by 5%. In future, we plan to extend our work to a comprehensive policy engineering framework that includes extracting ABAC policies from ACP sentences.

References

1. Apache pdfbox. https://pdfbox.apache.org/index.html
2. Federal information security management act of 2002. Title III of the E-Government Act of 2002 (2002)
3. Ibm course registration requirements (2004)
4. De Marneffe, M.C., MacCartney, B., Manning, C.D., et al.: Generating typed dependency parses from phrase structure parses. In: Proceedings of LREC. vol. 6, pp. 449–454. Genoa (2006)
5. Dolan, B., Brockett, C., Quirk, C.: Microsoft research paraphrase corpus. Retrieved March 29, 2008 (2005)
6. Forman, G., Cohen, I.: Learning from little: Comparison of classifiers given little training. In: European Conference on Principles of Data Mining and Knowledge Discovery. pp. 161–172. Springer (2004)
7. Gal-Oz, N., Gonen, Y., Yahalom, R., Gudes, E., Rozenberg, B., Shmueli, E.: Mining roles from web application usage patterns. In: International Conference on Trust, Privacy and Security in Digital Business. pp. 125–137. Springer (2011)
8. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD explorations newsletter 11(1), 10–18 (2009)
9. Hu, N., Bradford, P.G., Liu, J.: Applying role based access control and genetic algorithms to insider threat detection. In: Proceedings of the 44th annual Southeast regional conference. pp. 790–791. ACM (2006)
10. 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)
11. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. Machine learning: ECML-98 pp. 137–142 (1998)
12. Kong, D., Cen, L., Jin, H.: Autoreb: Automatically understanding the review-to-behavior fidelity in android applications. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 530–541. ACM (2015)
13. Lu, X.: Automatic analysis of syntactic complexity in second language writing. International Journal of Corpus Linguistics 15(4), 474–496 (2010)
14. Manning, C.D., Raghavan, P., Schütze, H.: Probabilistic information retrieval. Introduction to Information Retrieval pp. 220–235 (2009)
15. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit. In: ACL (System Demonstrations). pp. 55–60 (2014)

16. Medvet, E., Bartoli, A., Carminati, B., Ferrari, E.: Evolutionary inference of attribute-based access control policies. In: International Conference on Evolutionary Multi-Criterion Optimization. pp. 351–365. Springer (2015)
17. Meneely, A., Smith, B., Williams, L.: itrust electronic health care system: A case study (2011)
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 22th ACM Symposium on Access Control Models and Technologies. ACM (2017)
19. Narouei, M., Takabi, H.: Automatic top-down role engineering framework using natural language processing techniques. In: IFIP International Conference on Information Security Theory and Practice. pp. 137–152. Springer (2015)
20. 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)
21. Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. Advances in neural information processing systems 2, 841–848 (2002)
22. Ortega, L.: Syntactic complexity measures and their relationship to l2 proficiency: A research synthesis of college-level l2 writing. Applied linguistics 24(4), 492–518 (2003)
23. Ponzetto, S.P., Strube, M.: Knowledge derived from wikipedia for computing semantic relatedness. J. Artif. Intell. Res.(JAIR) 30, 181–212 (2007)
24. Rus, V., Lintean, M.C., Banjade, R., Niraula, N.B., Stefanescu, D.: Semilar: The semantic similarity toolkit. In: ACL (Conference System Demonstrations). pp. 163–168. Citeseer (2013)
25. 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)
26. Slankas, J.B.: Implementing database access control policy from unconstrained natural language text (2015)
27. Turney, P.D.: Mining the web for synonyms: Pmi-ir versus lsa on toefl. In: European Conference on Machine Learning. pp. 491–502. Springer (2001)
28. Van De Stadt, R.: Cyberchair: A web-based groupware application to facilitate the paper reviewing process. arXiv preprint arXiv:1206.1833 (2012)
29. Wolfe-Quintero, K., Inagaki, S., Kim, H.Y.: Second language development in writing: Measures of fluency, accuracy, & complexity. No. 17, University of Hawaii Press (1998)
30. 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)
31. Xu, Z., Stoller, S.D.: Mining attribute-based access control policies from logs. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 276–291. Springer (2014)
32. Xu, Z., Stoller, S.D.: Mining attribute-based access control policies. IEEE Transactions on Dependable and Secure Computing 12(5), 533–545 (2015)