Relation Extraction for Inferring Access Control Rules from Natural Language Artifacts

John Slankas, Xusheng Xiao, Laurie Williams
North Carolina State University
890 Oval Drive
Raleigh, North Carolina, 27695, USA
{john.slankas, xxiao2, laurie_williams}@ncsu.edu

Tao Xie
University of Illinois, Urbana-Champaign
201 N. Goodwin Ave
Urbana, Illinois, 61801, USA
taoxie@illinois.edu

ABSTRACT
With over forty years of use and refinement, access control, often in the form of access control rules (ACRs), continues to be a significant control mechanism for information security. However, ACRs are typically either buried within existing natural language (NL) artifacts or elicited from subject matter experts. To address the first situation, *our research goal is to aid developers who implement ACRs by inferring ACRs from NL artifacts.* To aid in rule inference, we propose an approach that extracts relations (i.e., the relationship among two or more items) from NL artifacts such as requirements documents. Unlike existing approaches, our approach combines techniques from information extraction and machine learning. We develop an iterative algorithm to discover patterns that represent ACRs in sentences. We seed this algorithm with frequently occurring nouns matching a subject–action–resource pattern throughout a document. The algorithm then searches for additional combinations of those nouns to discover additional patterns. We evaluate our approach on documents from three systems in three domains: conference management, education, and healthcare. Our evaluation results show that ACRs exist in 47% of the sentences, and our approach effectively identifies those ACR sentences with a precision of 81% and recall of 65%; our approach extracts ACRs from those identified ACR sentences with an average precision of 76% and an average recall of 49%.

Categories and Subject Descriptors
K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms
Documentation, Reliability, Security, Verification.

Keywords
Security, access control, classification, natural language parsing

1. INTRODUCTION
With over forty years of use and refinement, access control, often in the form of access control rules (ACRs), remains a significant and widely-used control mechanism for information security. ACRs regulate who can perform specific actions on specific resources within a software-intensive system and are considered a critical component to ensure both confidentiality and integrity [26]. Despite access control's widespread usage, systems continue to have vulnerabilities of incorrectly implementing ACRs. In the 2011 CWE/SANS Top 25 Most Dangerous Software Errors [1], 30% of the errors directly relate to access control.

Annual Computer Security Applications Conference '14, December 8–12, 2014, New Orleans, LA, USA.

To meet security requirements, ACRs need to be complete, consistent, and correct [37]. Analysts must manage ACRs such that they can be evaluated against these three attributes. Existing project artifacts for systems such as use cases, requirements documents, and user manuals often capture the intended ACRs for the systems. However, manually sifting through these existing artifacts to extract the buried ACRs can be a tedious, time-consuming, and error-prone endeavor.

To address this issue, various researchers have proposed approaches for extracting ACRs from natural language (NL) artifacts. These approaches leverage techniques such as controlled natural languages (CNLs) [2, 14, 28, 29], manual analysis [6, 11], and natural language processing (NLP) [30, 35]. Each of these techniques has its own strengths and weaknesses. For example, manual analysis typically produces the most accurate results, but at the cost of requiring more skilled human evaluators and time. Using CNLs produces comprehensive results as CNLs are designed to minimize the inherent ambiguity and complexity of NL [28]. However, using CNLs typically requires specialized authoring tools and conversion of pre-existing documents [29]. NLP techniques often require less manual effort than other techniques and can work on existing documents. However, NLP techniques tend to produce less accurate results than other techniques [15].

Our research goal is to aid developers who implement ACRs by inferring ACRs from NL artifacts.

We propose a novel approach, Access Control Rule Extraction (ACRE), to allow organizations to use existing, unconstrained NL texts such as requirements documents for inferring ACRs. ACRE combines NLP, information extraction (IE), and machine learning (ML) techniques. Internally, ACRE represents NL sentences as a type dependency parse graph [22] that shows words as vertices and relationships between words as edges. Figure 1 shows the graph for the sentence "The user enters a grade for each student".

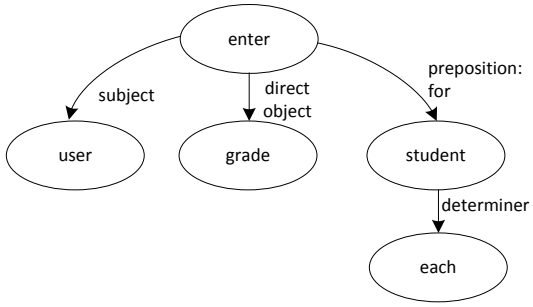

Figure 1. Type Dependency Graph

ACRE then searches these graphs for a basic pattern of *noun (subject)–verb–noun (direct object)*. ACRE creates a frequency table of all of the discovered subjects and direct objects. Any subject or direct object that occurs more times than the median count is assumed to a legitimate subject or resource (direct object) for the system that the requirements documents are for. ACRE then searches all combinations of the discovered subjects and resources within the graphs. For each found combination, the approach extracts the minimal spanning tree (MST) and assumes that any verb within the tree corresponds to the action. ACRE adds the extracted MST to a set of valid ACR patterns. ACRE then expands the pattern set by allowing a subject or resource node within the pattern to match any word of the same part of speech in other sentence graphs (i.e., "wildcarded"). ACRE then searches all of the sentence graphs for instances of the ACR patterns. The found instances are assumed to represent ACRs and the elements of the ACR are then extracted. ACRE iterates these steps, searching for additional patterns until no more patterns can be found in the sentences. To minimize incorrect patterns from being created and applied, ACRE constructs a naïve Bayes classifier to accept or reject patterns based upon domain-independent features found in other documents.

ACRE builds upon various concepts in the Text2Policy approach proposed by Xiao et al. [35]. However, ACRE incorporates IE and ML techniques to automatically learn new ACR patterns, whereas Text2Policy is limited by a fixed set of manually predefined ACR patterns.

This paper makes the following main contributions:

- Bootstrapping mechanism to seed and iteratively discover ACR patterns in NL text.
- Approach and supporting tool (built upon the combination of NLP, IE, and ML techniques) to extract ACRs.
- Labelled data set of identified ACRs and sentences[1].
- Evaluation of ACRE against documents from systems in three domains: conference management, education, and healthcare; Evaluation results showing that ACRE effectively identifies ACRs sentences with a precision of 81% and a recall of 65%, and ACRE extracts ACRs from those identified ACR sentences with an average precision of 76% and an average recall of 49%.

2. Background

This section provides background with regards to NLP, IE, ML, and the challenges faced by these techniques for extracting ACRs from NL documents.

2.1 Natural Language Processing (NLP)

NLP originates from the 1950s with the emergence of artificial intelligence research. Most notably, Chomsky's seminal work [5] demonstrated NLP's difficulty. Over 50 years later, perfect solutions do not exist for the vast majority of NLP tasks. NLP research in the 1960s and 1970s largely followed a rationalist approach with hand-built rules and grammars dominating the field [15]. In the 1980s, researchers returned to an empirical approach for NLP through the emergence of probabilistic techniques and large scale sets of annotated text [15]. These probabilistic techniques form the foundation of modern statistical parsers that represent the current state of the art in NLP [31]. Modern parsers include seven main tasks: tokenization, sentence segmentation, part-of-speech (POS) tagging, lemmatization, named-entity recognition, syntactic parsing, and coreference resolution.

Tokenization involves detecting individual words, punctuation, and other items from the text. In many situations, tokenization is straightforward in the English language due to spaces and punctuation marks; however, abbreviations, contractions, and other structures exist to complicate tokenization. Furthermore, technical documents may have structures (such as Java package names) that generally-trained tokenizers (lexical analyzers) cannot handle.

Sentence segmentation identifies sentence boundaries and then splits the tokens found in tokenization into groups.

POS tagging identifies the part-of-speech tags (e.g., noun, verbs, adjectives) for each token. The Penn Treebank [27] contains 36 different tags. The current state of the art achieves 97.3% accuracy for individual tokens and a "modest" 57% for accuracy for an entire sentence [21].

Lemmatization generates the common root word for a group of related words. For instance, sang, sung, and sings are all forms of a common lemma "sing". This common root differs from a stem, which is the root of a word after a suffix has been stripped [15]. Stems are produced through rule-drive techniques to derive base forms of words without taking into account the word's context or part of speech. In contrast, lemmatization takes into account the word's part of speech and other information sources such as a vocabulary to derive the base form of a word [20]. The state of the art achieves around 99% accuracy for the English language [9].

Named-entity recognition (NER) seeks to classify phrases into entity types (e.g., people, organizations, locations, times, vehicles, events) from text [15]. Real-world NER architectures use a combination of high-precision rules, probabilistic matching, and machine learning techniques [15]. The state of the art for the NER general task (the CoNLL-2003 shared task) has a F_1 score of .89% [18].

Syntactic parsing assigns a parse-tree structure to a sentence [15]. The tree structure provides a basis for other tasks within NLP such as question and answer, IE extraction, and translation. The state of the art parsers have an F_1 score of 90.4% [31].

Coreference resolution identifies whether or not two expressions in a document refer to the same identity. A common subset of this problem occurs within extracting ACRs from NL texts in that pronouns must be resolved to their antecedents (the actual role or resource). The state of the art has a 78% accuracy for the CoNLL-2012 Shared Task [25]. Kennedy and Boguraev [16] introduced an algorithm to resolve pronoun anaphora resolution (match the correct noun to a pronoun) that does not require parsing and achieves a 75% accuracy on their test set.

2.2 Information Extraction (IE)

IE creates structured data from text [15]. Common IE tasks include named-entity recognition, reference resolution, relation extraction (RE) and event extraction. A relation expresses the relationship among two or more items. Common relation types include "is-a" and "part-of". For example, "a doctor is a licensed healthcare practitioner (LHCP)" is represented by *is_a(doctor, LHCP)* and "medical records contain family history" is represented by *contains(medical record, family history)*. Similarly, we can have relation *writes(doctor, prescription)* to indicate that a doctor can write a prescription. The IE field has advanced largely due to the investments by the United States government through challenges sponsored by DARPA [4] and NIST[2]. State-of-the-art systems for RE in the sponsored challenges typically have around 85% precision and 70% recall [24]. In a task similar to ACR extraction, Zhu et al. [36] reported a

[1] https://sites.google.com/site/AccessControlRuleExtraction

[2] http://www.itl.nist.gov/iad/mig//tests/ace/

0.742 F_1 score for extracting medical semantic relations (*problem,tests,treatments*) from clinical texts.

ACR extraction is most similar to the RE task in that the relation (action) between the subject and resource implies much of the ACR within a sentence. However, ACR extraction differs from most RE tasks in that it is not constrained by small, fixed sets of binary relations [15]. Additionally, ACR extraction needs to infer the appropriate permissions based upon the identified relation (action). The permission may be further constrained by other features within the sentences such as negativity or access limitations to just a particular person or role.

Another IE technique is shallow parsing (semantic role labeling). This technique involves identifying the different predicates (phrases) in a sentence along with the appropriate role for each phrase [3, 15]. The labelled roles then constitute the subject, action, and resource for an ACR.

2.3 Machine Learning (ML)

ML allows us to employ an inductive technique for extracting ACRs from NL rather than specifying specific extraction rules. To decide whether or not a sentence contains an ACR, we construct a *k*-nearest neighbor classifier (*k*-NN) [10], which is a supervised learning algorithm. A *k*-NN classifier works by classifying a test item based upon which items previously classified are closest to the current test item. The classifier finds the *k* nearest "neighbors" and returns a majority vote of those neighbors to classify the test item. A distance metric determines the closeness between two items. Euclidean distance often serves as a metric for numerical attributes. For nominal values, the distance is binary – zero if the values are the same or one if the values differ. *k*-NN classifiers may use custom distance functions specific to the current problem. Advantages of *k*-NN classifiers include the ability to incrementally learn as new items are classified, to classify multiple types of data, and to handle a large number of item attributes. The primary drawback of *k*-NN classifiers involves high algorithm runtime cost; if the classifiers have *n* items stored, classification takes $O(n)$ time.

We also construct a naïve Bayes (NB) classifier to decide whether or not a generated pattern is valid. These classifiers work by selecting a class with the highest probability from a set of trained data sets given a set of features [10]. Fundamentally, it assumes that each feature of a class exists independently of other features and the probabilities are derived from the counts of features occurring in each class. Despite the simplification, the technique performs effectively in real-world problems.

We selected both classifiers after evaluating multiple types of classifiers with different features. The classification performance on the test data was used as the selection criteria. As aside, we found that a tree classifier had the highest performance for the training data in classifying patterns, but a lower performance score for the test data as the tree classifier overfits the training data.

2.4 Challenges

Many significant, complex problems exist for NLP, IE, and ML; these problems carry forward into ACR extraction.

Ambiguity issues appear in many contexts, including what meaning words have. For example, the sentence "the bank collapsed yesterday" creates confusion as one does not know whether the "bank" was a financial institution or the side of levee. Systems must disambiguate such terms based upon the context – within the sentence itself or other sentences in the document. Similarly, the same word can create issues for ACR extraction. For example, "the patient *enters* his zip code to find the closest physician" and "the administrator *enters* a new patient" have separate permissions implied for the verb "enter". In the first case, the patient searches for physicians based upon a zip code so a read permission is necessary. In the second case, the administrator effectively creates a new patient in the system and, hence, some form of a create permission is necessary.

Synonyms for the same term are another frequent source of ambiguity in NL texts. For example, "professor" and "instructor" may refer to the same role in sentences S2 and S4 in Figure 2.

Negativity, denying users access to a specific action and/or resource, is a critical problem for access control. In certain cases, users may be granted access to part of a record, but then explicitly denied from reading other parts of that record. Negativity can appear in multiple ways within sentences: negative determiners (e.g., no, zero, neither), adjectives (e.g., unable), nouns (e.g., none, nothing), verbs with negative connotation (e.g., stop, prohibit), and adverbs (e.g., never) [12]. All such possibilities should be considered, and the identified negativity should be tied back to the appropriate subject, action, or resource.

Resolution issues also appear in ACR extraction. One common situation is the appearance of pronouns in place of the actual subjects or resources. A specific case, anaphora resolution, exists when a pronoun or other phrase (e.g., that, there) refers back to a previously occurring noun or entity (the antecedent). A similar resolution situation occurs when generic terms such as "data", "entries", and "records" appear in text. Another situation occurs when the system appears to be the subject (S3 in Figure 1) when the actual actor can be identified from a prior sentence.

S1: The system displays a list of courses taught by the professor.
S2: The professor selects a course.
S3: The system displays a list of enrolled students for the course.
S4: The instructor can enter a grade for each student.

Figure 2. Sample ACP Sentences

3. Access Control Rule Extraction (ACRE)

This section details ACRE – our approach to extracting ACRs from NL text.

3.1 ACR Representation

Internally, ACRE represents sentences with a dependency graph [22] as depicted in Figure 3 for the sentence "a nurse can order a lab procedure for a patient." In Section 3.2.2, we discuss how these graphs are produced. Each vertex represents a word from the sentence along with the word's part of speech. In the figure, "NN" represents a noun, "VB" represents a verb, and "MD" represents a modal verb. Edges represent the grammatical relationship between two words. For instance, "nurse" functions as the nominal subject (nsubj) for "order", and "lab procedure" is the direct object (dobj) to be ordered. The indicators correspond to the subject("S"), action("A"), resource("R") typically defined within an ACR. Dependency graphs can be considered as trees in most situations and are typically rooted by the sentence's main verb. When conjunctions are present, vertices may have multiple parents, and thus the structure needs to be treated as a graph.

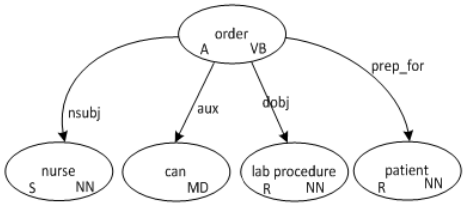

Figure 3. ACRE Sentence Representation

To represent an ACR, we use the pattern presented in Figure 4. A defines the overall ACR. s contains an order set of vertices that comprise the subject of a rule. Similarly, a and r represent the

action and resource, respectively. n contains the vertex representing negativity if required for the rule. If the rule should be limited to a particular subject s, l contains the indicating vertex. c contains additional vertices required to provide context to a given action for a set of permissions. H represents the subgraph of a sentence's dependency graph that contains the vertices and necessary edges to connect all of the vertices listed in s, a, r, n, l, c. p represents the permissions typically associated with an action. We limit permissions to have the values of "create", "retrieve", "update", and "delete" as we are primarily concerned with controlling the ability to view and manipulate data in systems. We do use "execute" for permissions that do not map to one or more of the four preceding permissions. From the example in Figure 3, we define the two rules in Figure 5.

$$A(\{s\},\{a\},\{r\},[n],[l],\{c\},H,p)$$

Figure 4. ACR Representation

$A((nurse),(order),(lab\ procedure),(\ ),(\ ),(V: nurse, order, lab\ procedure;$
$E: (order, nurse, nsubj); (order, lab\ procedure, dobj)\ ), create)$
$A((nurse),(order),(patient),(\ ),(\ ),(V: nurse, order, patient;$
$E: (order, nurse, nsubj); (order, patient, prep\_for)\ ), read)$

Figure 5. Extracted ACRs

Situations exist in which not all of the ACR elements may be present within a single sentence. ACRE allows for only subjects to be missing. While users may manually identify ACRs with missing resources, ACRE has no support to find such patterns. Developers would be pointed to the surrounding sentences to finish defining the ACRs outside of the automated process.

3.2 ACRE Approach Details

The ACRE approach consists of five main steps:

1. Preprocess text documents
2. Produce dependency graphs
3. Classify each sentence as access control or not
4. Extract access control elements
5. Validate access control rules

3.2.1 Step 1: Preprocess Text Documents

In the approach, we first read the entire text into the tool built to support the approach. We separate the input into lines by either a carriage return or by periods at the end of sentences. Next, we apply a concise grammar [30] to label each token to a specific type (title, list element, list start, normal). By identifying specific types of sentences, we can increase the classification performance. Specifically, we found that section titles typically do not contain ACRs, and we design the mechanism of scoring sentence similarity to compare titles with only other titles within the document. We also found that list items require more context than just the specific item itself and process each list item combined with the start of the list.

3.2.2 Step 2: Produce Dependency Graphs

In our approach, after identifying the different sentence types, we parse each line (sentence) using the Stanford Natural Language Parser[3] and output a graph in the Stanford Type Dependency Representation (STDR) [22]. While the Stanford Parser has several output formats available, we choose the STDR because it incorporates the sentence's syntactic information in a concise and usable format and captures the grammatical relationship between words. Dependency parse trees have become a critical component for many semantic role labeling systems as the NLP community evolved in the early 2000s from a strict constituent-based (i.e., shallow parsing) systems [33]. From the STDR generated by the parser, we create the sentence representation (SR) since we need to track additional attributes for the sentence and for each word.

3.2.3 Step 3: Classify Each Sentence as Access Control or Not

Next, a k-NN classifier classifies whether a sentence contains an ACR. If the sentence does not express an ACR, we perform no further analysis on it. The k-NN classifier works by taking a majority vote of the existing classifications of the k nearest neighbors to the item under consideration. The classifier uses an adapted version of the Levenshtein distance [19] as the distance metric. Rather than using the resulting number of edits to transform one string into another as the Levenshtein distance does, our adapted distance metric computes based on the number of word transformations to change one sentence into another.

Although other machine learning algorithms can provide similar performance to a k-NN classifier, the k-NN classifier provides easier interpretation of the results for analysts since they can see a ranked list of similar sentences and associated classifications.

Once we determine that the sentence contains an ACR, the user may review the determination and correct it if necessary within the tool. Figure 6 shows a screenshot of the tool's user interface. The top table contains the document with individual columns to display the line number, sentence type, assigned classification, and completion status, assigned cluster (groups of similar sentences, optional functionality), and the sentence themselves. The dialog in the lower left allows users to review and manually enter or correct ACRs (discussed in the next section). The area in the lower right displays the SR.

3.2.4 Step 4: Extract Access Control Elements

Next, we need to extract the subject, action, and resource elements from the SR. We construct a relation extraction algorithm for the identification of ACR elements and subsequent extraction of the ACR. The algorithm follows a well-known bootstrapping technique [7], but has been adapted specifically for ACR extraction. The basic concept is to start with a small, well-known set of ACR patterns and then expand those patterns to find other, closely related patterns.

To initialize the algorithm (presented in Figure 8), we seed a set of ten basic ACR patterns with each pattern consisting of just three vertices as shown in Figure 7. Each pattern is the same, except a different verb[4] is used for a "Specific Action". Wildcards are used to match any noun in sentences containing the pattern. We initially chose the words "create", "retrieve", "update", and "delete" because the words are commonly associated with viewing and manipulating data. We then examined the frequencies of all verbs within the test documentation and chose to add more verbs associated with data and appearing with high frequencies within the document. Based upon the application domain or other documents, users may choose a different set of starting actions. From these patterns, we match all occurrences of the subjects and resources within the document along with their associated frequency counts. From the counts, we compute the median values for the subjects and resources. We then assume any word that occurs more than the median belongs to the application domain. Without a threshold, the potential for misidentified subjects and resources is much greater as any word matching the pattern would be accepted.

The subjects and resources are then stored in a list of known subjects and resources. From this listing, we then search the documents to see whether any subject exists along with any

[3] http://www-nlp.stanford.edu/software/corenlp.shtml

[4] create, retrieve, update, delete, edit, view, modify, enter, select

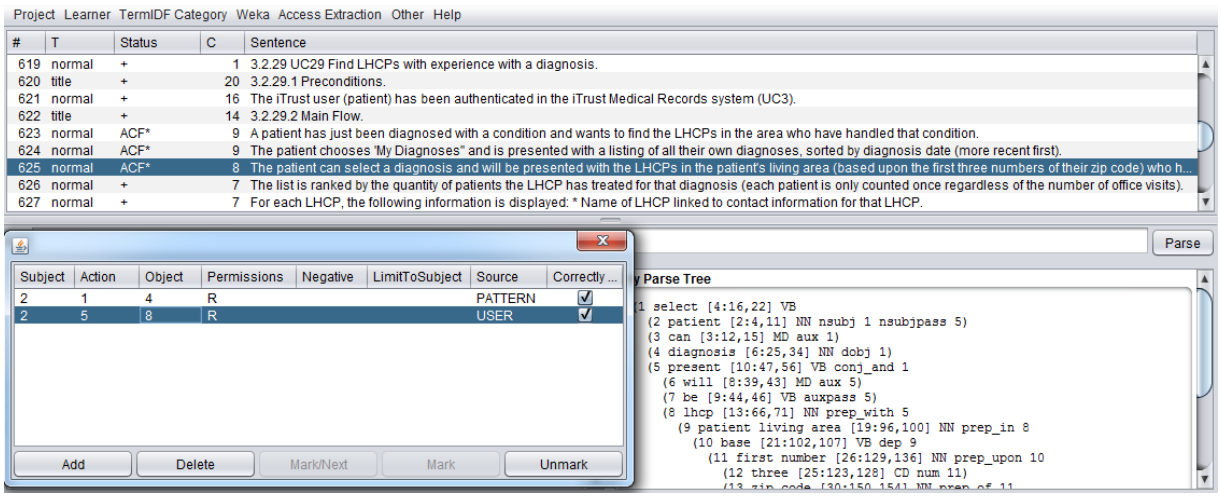

Figure 6. Tool Screenshot of Access Control Rule Extraction (ACRE)

resource. For each sentence that does match the condition, we extract the dependency pattern between subject and resource vertices. We then assume that any verbs existing in that pattern are the actions. If more than one verb exists in the shortest path from the subject to the object, we combine the verbs, but use the last appearing verb when defining permissions. In the sentence, "the administrator chooses to create a new patient", we combine "choose" and "create" to "choose create" for the action. The subject would be "administrator" and the object would be "patient". We derive permissions for each pattern by finding the closest synonym (via WordNet[5]) that has an already defined permission.

Once we extract the pattern, we apply a series of transformations to extract additional patterns that may locate additional ACRs. Specifically, we transform patterns that have an active voice into passive voice and vice versa. We also transform the patterns to assume conjunctions may exist for two or more subjects, two or more actions, and two or more resources. To find additional subjects and resources, we apply wildcards to the identified subject or resource vertices. Only the subject or the resource vertex is wildcarded to minimize semantic drift in bringing in unrelated sentences or patterns to access control.

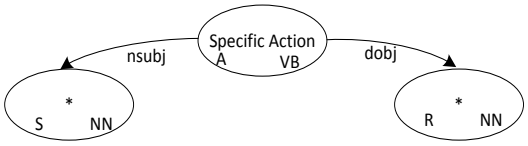

Figure 7. Basic ACR Seed Pattern

Furthermore, to ensure that the patterns are appropriate, we apply a naïve Bayes classifier to judge whether an identified pattern is appropriate. To detect the features used by the classifier, we use a forward, stepwise selection model [10]. From this model, we found the pattern itself (the POS tags and relationships between vertices) to have the biggest influence. After that, the relationships to the resources and subjects had the next largest influence. After that, the identified parts of speech for the resources and subjects improved classification performance slightly. We did not use any further features (size, specific words, use of wildcards) since we found those features began to decrease the classifier's performance.

From the pattern set, we then search the documents for sentences matching one or more patterns. Once we find any match, we check to see whether other patterns match the same sentence. If more than one pattern matches, and one pattern is contained within another pattern, we discard the former as the latter pattern provides a more specific match. Additionally, we check the matched sentences for any children vertices (of the matched pattern) that imply negativity or subject limitation (i.e., we check whether a relevant indicator exists just outside of the matched pattern).

The extracted ACR is then stored in a list for validation and output to the user. Any new subjects or resources are then added to the list of known subjects and resources. If newly discovered subjects or resources exist, then the algorithm iterates until no new items or patterns are discovered. Once the algorithm converges, the user may inject two additional patterns into the process to find more ACRs. Similar to the Basic ACR Seed Pattern in Figure 7, the first injected pattern allows any pronoun to be a subject within an ACR pattern. The second pattern searches for only actions and resources, leaving resolution of the subjects to another algorithm. The injection occurs after the algorithm initially converges to avoid spurious matches that would occur if the patterns were injected at the start of the algorithm. Additionally, the user may manually identify ACR patterns through identifying ACRs in the sentences. The information from these patterns is fed into the algorithm to search for additional extracted elements.

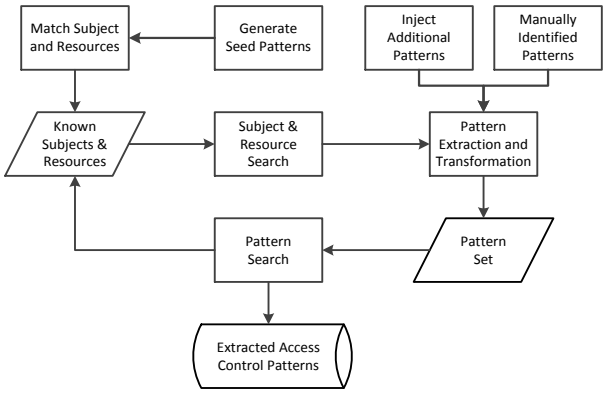

Figure 8. ACR Extraction Overview

[5] http://wordnet.princeton.edu/

Table 1. Study Documents						
Document	**Abbreviation**	**Domain**	**Number of Sentences**	**Number of ACR Sentences**	**Number of ACRs**	**Fleiss' Kappa**
iTrust for ACRE	iTrust_acre	Healthcare	1160	550	2274	0.58
iTrust for Text2Policy	iTrust_t2p	Healthcare	471	418[10]	1070	0.73
IBM Course Management	IBM_cm	Education	401	169	375	0.82
CyberChair	Cyberchair	Conference Mgmt	303	139	386	0.71
Collected ACP Documents	Collected	Multiple	142	114	258	n/a

3.2.5 Step 5: Validate Access Control Rules

In this step, the tool checks for coverage and conflicts within the extracted ACRs. Coverage is reported as the measurement for each subject as to the number of identified resources that it has ACRs identified. As we assume a default of no-access, 100% coverage is not required. However, low coverage values may indicate a need for further ACRs. Conflicts occur within ACRE when a specific subject has been both granted permission to a specific resource and restricted for the same permission on the same resource. Such conflicts may arise due to rule extraction in multiple locations or the use of a limiter to restrict access to a specific subject.

4. Study Methodology

This section presents the methodology for collecting the study documents, creating the study oracle, and running the analysis.

4.1 Study Documents

As access control widely exists in numerous domains for software systems, we chose multiple domains for the evaluation. We selected documents from the electronic healthcare, educational, and conference management domains. Additionally, to compare results to prior work, we included Xiao et al.'s study documents [35]. Table 1 lists the study documents.

For the electronic health care domain, we selected iTrust[6] [23]. The requirements consist of 40 use cases plus additional non-functional requirements, constraints, and a glossary. We used two versions of this document. The first (iTrust for ACRE) was extracted directly from the project's wiki[7] while the second (iTrust for Text2Policy) was taken from the documentation[8] used by Xiao et al. [35]. The first version more closely matches specifications used in industrial settings in that it has separate sections for introduction, glossary, non-functional requirements, and other materials. The second version includes only the use cases themselves and simplifies complex sentences to be more consistent with the rules of their parser [35]. For the educational domain, we took the eight use cases from the IBM Course Registration System [13] used in a prior research study [34]. For the conference management system, we used documents from CyberChair[9] [32], which has been used by over 475 different conferences and workshops. We also included a combined document of 114 sentences with ACRs that Xiao et al. [35] collected.

4.2 Study Oracle

To train the classifiers used within ACRE and to evaluate its performance, we developed a study oracle. To create the oracle, the first author followed the following steps:

1. Convert the document into a "text-only" format.
2. Correct the resulting text file to account for improper line breaks and other formatting issues.
3. Import the document into the ACRE tool.
4. Mark each sentence as to whether or not it is an ACR sentence.

After the classification was complete, we validated the classification through multiple ways. First, we created clusters of related sentences. We then compared the classifications within each cluster and investigated further those sentences that did not have the same classification as other sentences in the group. Additionally, as we classified each sentence, we had access to the neighbors contained within the *k*-NN classifier. This way allowed for more rapid manual classification by suggesting initial classification that we could then verify or correct as deemed necessary. Additionally, any discrepancies in the predicted classification could be easily traced back to the source sentences.

We then had two other researchers manually classify each document. We then computed the Fleiss' kappa [7] for each document (see Table 1). From Landis' and Koch's guidelines [17], scores between 0.41 and 0.60 indicate moderate agreement, while those between 0.61 and 0.80 indicate substantial agreement. 0.81 to 1.00 is considered almost perfect agreement. The three individuals then discussed the differences and collectively choose the appropriate classification for each sentence with disagreements.

The first author then manually identified each ACR within the documents. Once those rules were identified, another researcher then validated a random sampling of 20% of the ACRs from the document "iTrust for ACRE". To ensure that the reviewer diligently examined each set of tuples, we injected five "mistakes". The reviewer made 29 corrections out of the 359 ACRs and found 80% of the injected "mistakes".

To create the initial classifications of the five documents, the first author spent six hours to classify the 2,477 sentences as access control or not. Identifying the specific access tuples took additional 62 hours for the five documents.

4.3 Study Procedure

Once the oracle has been created, we ran the ACRE Tool to produce several reports to pull out details to examine properties of sentences with access control. To evaluate how well we identify sentences with ACRs, we ran the *k*-NN classifier on a combined document of the iTrust ACRE requirements, IBM Course Management, and CyberChair. We also report results on each of the five documents being individually classified. Each document was tested with stratified *n*-fold cross-validation and computed the precision, recall, and F_1 measure. With the *n*-fold cross-validation, data is randomly partitioned into *n* folds based upon each fold of approximately equal size and equal response classification. For each fold, the classifier is trained on the remaining folds and then

[6] http://agile.csc.ncsu.edu/iTrust/

[7] http://agile.csc.ncsu.edu/iTrust/wiki/doku.php?id=requirements

[8] https://sites.google.com/site/text2policy/

[9] http://www.borbala.com/cyberchair/

[10] Xiao et al. [35] reported 448 sentences with 117 containing access control for the same document evaluated from their web site (https://sites.google.com/site/text2policy/). In this work, we marked a sentence as containing an access control rule(s) when an actor performed an action with regards to some resource within the sentence. In the prior work [35], sentences with access control had to follow one of four patterns (Table 6).

Table 3. Metrics by Document					
	iTrust_acre	iTrust_t2p	IBM CM	CyberChair	Collected
Text2Policy Pattern – Modal Verb	210	130	46	71	93
Text2Policy Pattern – Passive voice w/ to Infinitive	66	21	10	39	9
Text2Policy Pattern – Access Expression	32	7	5	1	18
Text2Policy Pattern – Ability Expression	45	21	14	11	3
Number of sentences with multiple types of ACRs	383	146	77	105	36
Number of patterns appearing once or twice	680	173	162	184	97
ACRs with ambiguous subjects (e.g. "system", "user", etc.)	193	119	139	1	13
ACRs with blank subjects	557	206	29	187	5
ACRs with pronouns as subjects	109	28	5	11	11
ACRs with ambiguous objects (e.g., entry, list, name,etc.)	422	228	45	47	34

the contents of the fold are used to test the classifier. The *n* results are then averaged to produce a single result. We follow Han et al.'s recommendation [10] and use 10 as the value for *n* as this value helps produce relatively low bias and variance. The cross-validation ensures that all sentences are used for training and that each sentence is tested just once. We also report the results of using the individual documents as folds within the combined document. As it is not necessarily feasible to have a trained classifier readily available, we also evaluate the amount of work necessary to classify a document from scratch in terms of the performance when classifying the rest of the document.

In the final phase of the study, we created the naïve Bayes classifier using a document-fold way (i.e., training the classifier with all of the documents being evaluated except for one and then repeating the experiment until all documents have been tested). We then ran the ACRE process to extract the ACRs. The extracted ACRs were compared against the study oracle to determine the accuracy of the results.

4.4 Evaluation Criteria

To evaluate results, we use recall, precision, and the F_1 measure. Recall measures how many of the ACR sentences we identified from all of the sentences. Precision measures how well we identified the ACR sentences in looking at classification mistakes. To compute these values, we categorize the classifier's predictions into four categories. True positives (TP) are correct predictions. True negatives (TN) are cases where we correctly predicted that a sentence was not an ACR sentence. False positives (FP) are cases where we mistakenly identify a sentence as an ACR sentence when it is not. False negatives (FN) occur when we fail to correctly predict an actual ACR sentence. From these values, we define precision (P) as the proportion of correctly predicted classifications against all predictions against the classification under test: $P = TP/(TP + FP)$. We define recall as the proportion of classifications found for the current classification under test: R = TP/(TP+FN). The F_1measure is the harmonic mean of precision and recall, giving an equal weight to both elements: $F_1 = 2 \times \frac{P \times R}{P+R}$. From an access control perspective, high values for both precision and recall are desired. Lower precision implies that the approach could more likely identify non-ACR sentences as ACR sentences. Lower recall implies that the approach could more likely miss ACR sentences.

5. Evaluation

We present and answer our research questions in this section.

5.1 Access Control Sentence Analysis

RQ1: What patterns exist among ACR sentences?

For this research question, we explore different patterns that ACR sentences have. Xiao et al. reported four common sentence patterns for sentences with ACRs (see Table 6). In Table 2, we present the number of times we found these sentence patterns within our study documents. As we did not have access to their tool, we identified how often ACR sentences met one of their four patterns based upon when certain conditions were met. For their "Modal Verb in Main Verb Group" pattern, we checked whether a modal verb existed in an ACR sentence. For the "Passive Voice" pattern, we checked whether the sentence was in passive voice with "allow" and "to" appearing in the sentence. For the "access expression" pattern, we checked whether an ACR sentence had some form of "access" within it. Finally, for the "Ability Expression" pattern, we whether there existed some form of "access" within the sentence. Xiao et al. found that 85% of the ACRs followed these patterns in the document of iTrust for Text2Policy. The updated results derived by us showed that these patterns cover only 57% of the ACR sentences that we identified.

We also examined how frequently extracted ACRs pattern appear. Table 2 shows the top ACR patterns for all documents. The most common pattern (equivalent to ACRE's basic seed pattern) occurs approximately 14% for all ACRs. We also found a high occurrence of ACR patterns with a preposition in them.

Table 2. Top ACR Patterns	
Pattern	**Num. of Occurrences**
`(VB root(NN nsubj)(NN dobj))`	465 (14.1%)
`(VB root(NN nsubjpass))`	122 (3.7%)
`(VB root(NN nsubj)(NN prep))`	116 (3.5%)
`(VB root(NN dobj))`	72 (2.2%)
`(VB root(NN prep_%))`	63 (1.9%)

We then examined how frequently multiple types of ACRs could occur in one sentence. For example, S3 in Figure 2 has multiple forms of ACRs to be extracted. We found multiple ACRs in approximately 33% of all ACR sentences. Multiple forms of access being tied to the same action can also cause issues with permissions since different permissions may be necessary.

RQ2: How frequently do different forms of ambiguity occur in ACR sentences?

We also examined how frequently different forms of ambiguity (i.e., when a subject or object is not clearly defined in an extracted ACR), occur within the documents. We found that pronouns occurred as subjects in only 3.2% of the identified ACRs. "System" and "user" occurred as subjects in 11% of all ACRs. We also found that a subject could not be explicitly identified as a subject in 17.3% of the ACRs. We found very few instances of pronouns as objects. However, we did find that ambiguous terms (e.g., "list", "name", "record") occurred 21.5% in all of the ACRs. From these observations, a strong need exists to track actors from one sentence to another sentence to complete blank subjects or replace "system" and "user" occurrences. Similarly, an effective approach needs to be able to resolve ambiguity issues with objects. In most cases, this ambiguity resolution can be done by examining prepositions that relate directly to the ambiguous word. We also found that missing objects within ACRs only occurred in four times in all of the documents.

5.2 Identification of ACR Sentences

RQ3: How effectively does ACRE detect ACR sentences in terms of precision and recall?

Table 4 presents the results of running the classifier against each document individually with a ten-fold cross validation. We then evaluate the documents of iTrust for ACRE, IBM Course Management, CyberChair, and Collected ACP using both a 10-fold cross validation and a document-fold validation. For general use, the document-fold validation would most represent the performance as the user would use an existing classifier against a new document. For this result, ACRE had a precision of 81% and a recall of 65%. The scores for the documents of iTrust for Text2Policy and Collected ACP were abnormally high due to the high concentration of ACR sentences.

We also evaluated how ACRE would perform classifying sentences in which the user did not have a pre-trained classifier available. Figure 9 shows the classification performance for the documents based upon what percentage of the sentences in a document has been correctly classified by a user. For the document of iTrust for ACRE, the F_1 score is already above 80% after just 25% of the sentences in the document have been classified as having ACRs or not. For the sentences in the document of IBM Course Management, the performance shows two irregularities where performance decreases (at 10% and 35% completion). Based on examining the document, the first situation contains the glossary. The sentences contained here tend to have high similarity to other sentences later in the document, but do not have any corresponding user-based actions. As such, they do contain ACRs. They then cause false negatives as later sentences are parsed. In the second situation, the User Login Use Case was evaluated. As these sentences deal primarily with authentication, they do not contain ACRs. The documents for iTrust for Text2Policy and Collected ACP were not placed into the graph since the overall percentages of ACR sentences are so high that there is no practical change to the classification performance.

Table 4. Identification of ACR Sentences

Document	Precision	Recall	F_1
iTrust for Text2Policy	96%	99%	98%
iTrust for ACRE	90%	86%	88%
IBM Course Management	83%	92%	87%
CyberChair	63%	64%	64%
Collected ACP	83%	96%	89%
10-fold validation	81%	84%	83%
Document-fold validation	81%	65%	72%

5.3 Access Control Rule (ACR) Extraction

RQ4: How effectively can the subject, action, and resources elements of ACRs be extracted from ACR sentences?

Table 5 presents the results of the bootstrapping algorithm on each of the documents. We train the naïve Bayes classifier by using the identified patterns in the other documents. The document of iTrust for Text2Policy is not used for training. Also, neither of the iTrust documents are used in the training set when evaluating one of those documents.

Table 5. ACR Extraction

	Precision	Recall	F_1
iTrust for Text2Policy	80%	75%	77%
iTrust for ACRE	75%	60%	67%
IBM Course Management	81%	62%	70%
CyberChair	75%	30%	43%
Collected ACP	68%	18%	29%

ACRE performed best on document of iTrust for Text2Policy. This document contained a number of subjects and resources repeated throughout the document. As complex sentences were split into multiple, simpler sentences, there were less complex patterns to discover. ACRE performed worst on the document of Collected ACP. This document contains ACR sentences extracted from 19 sources. As little repetition exists in sentence structure, subjects, and resources, ACRE performs poorly in finding the initial set of known subjects and resources as well as in expanding the patterns. For the document of Collected ACP, if we start the algorithm with a known list of resources and subjects (which can be easily obtained from a glossary or similar section), ACRE has a precision of 85%, a recall of 70%, and a F_1 of 77%. The document of CyberChair also demonstrated a very low recall. As this document was a combination of an introductory page as well as a conference paper, little repetition exists in the sentence structure. Bootstrapping from a known list of subjects and resources only slightly improves the recall to 39% as 117 of the missing 240 ACRs were covered by patterns with more than three nodes containing missing subjects.

When examining which features to use for the naïve Bayes classifier, we found that the patterns themselves produced 80% of the possible performance. The relationships to the subjects and the objects then made the next most noteworthy performance improvements. After these three features, adding more features into the classifier reduced the classifier's performance.

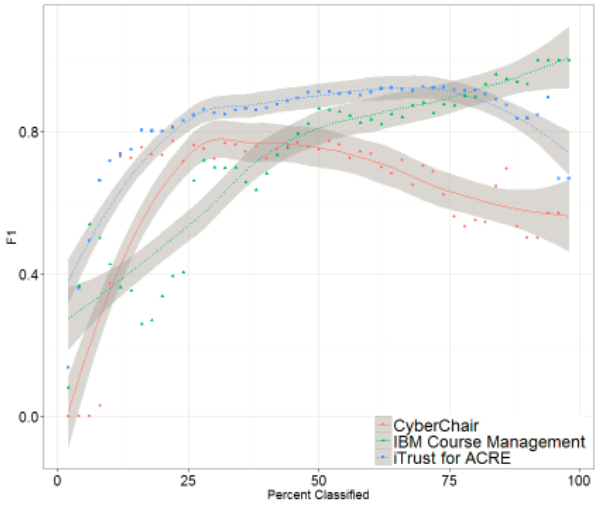

Figure 9. Classification Performance (F_1) by Completion %

6. Discussion and Future Work

Since our ACRE approach uses NLP techniques, ACRE and its supporting tool cannot extract information contained in images. With regards to ACRs, the bootstrapping mechanism does not take into account the presence of contextual information or conditions that may affect the generated ACRs. But the user can manually enter such information. The ACRE approach also requires that subjects and resources be identified as nouns and actions as verbs unless the user manually enters a rule. The approach also assumes that all necessary information for an ACR is contained within the same sentence. But it is feasible for elements of an ACR to exist in surrounding sentences of the corresponding ACR sentence.

Our approach currently does not handle resolution issues. These issues occur when a pronoun or generic term such as "system" or "data" is used in place of a descriptive term. In future work, we plan to incorporate resolution techniques to address such issues. We also plan to investigate how to search for larger ACR patterns in which the subject is missing.

The threat to external validity is mainly due to the representativeness of the subjects. To reduce the threat, we

evaluate our approach on documents from three domains. However, to further reduce the threat, additional evaluation needs to occur across multiple domains and applications. We surmise that the approach can work for other narrative-based texts, but "task/step-oriented" documents such as test scripts and user manuals would be less effective as the subject is often assumed throughout a series of steps. For such documents, we would need to study the use of "action–resource" pairs to generate patterns.

ACRE does require manual effort to setup the classifiers. We were able to identify whether or not a sentence contained an ACR at an average rate around one sentence per 9 seconds. However, this identification process is just making a simple yes or no decision and we had optimized an interface such that the user only had to press a single button per sentence. Considerably more effort is required to identify each ACR within a sentence. We identified the ACRs at an average rate of one ACR per 50 seconds. However, we found that the naïve-Bayes classifier for patterns could be used effectively across our documents and domains. In future work, we plan to study ways to reduce the workload in human annotation for NL classification tasks.

7. Related Work

This section presents related work in regards to controlled NLs and extraction of ACRs from NL artifacts.

7.1 Controlled Natural Language (CNL)

Approaches of controlled natural language (CNL) were proposed to convert NL to and from ACRs. Schwitter [28] defined a CNL as "an engineered subset of a natural language whose grammar and vocabulary have been restricted in a systematic way in order to reduce both ambiguity and complexity of full natural language." While a CNL provides consistent, semantic interpretations, a CNL limits authors to the defined grammar and typically requirse language-specific tools to stay within the language constraints. Previously created project documents cannot be used as inputs without pre-processing the documents into CNL-specific tools. Rules authored outside of the tools must conform to strictly limited grammars to be automatically parsed. Brodie et al. [2] used such approach in the SPARCLE Policy Workbench. By using their own NL parser and a controlled grammar, they effectively translated from NL into the formal rules. Inglesant et al. [14] demonstrated similar success with their tool, PERMIS, which used a role-based authorization model. However, they reported issues with users not comprehending the predefined "building blocks" imposed by using a CNL. Recently, Shi and Chadwick [29] presented their approach to authoring ACRs using a CNL. While they showed the improved usability of CNL interface, their approach was limited in the complexity of the rules that could be created since their supporting tool did not support conditions such as previous actions that must be taken before a user could access data. ACRE removes the CNL constraints, working against original, unconstrained texts.

7.2 Natural Language and Access Control

NL sources have been analyzed to infer and generate ACRs. Fernandez and Hawkins [6] presented a basic overview of extracting RBAC from use cases in 1997. Fontaine [8] proposed an approach based upon goal-based requirements engineering to extract authorization and obligation rules from NL texts into a policy language. He and Antón [11] proposed an approach to generate ACRs from NL based upon available project documents, database design, and existing rules. Using a series of heuristics, developers manually analyze the documents to find ACRs whereas our approach seeks to automatically extract ACRs.

7.3 Text2Policy

Xiao et al. proposed Text2Policy [35], for automated extraction of ACRs. Text2Policy accepts use cases in NL text as input and outputs the extracted ACRs in the eXtensible Access Control Markup Language (XACML) format. Text2Policy first uses shallow parsing techniques with finite state transducers to annotate sentences with "phrases, clauses, and grammatical functions of phrases such as subject, main verb, and object." From those annotations, Text2Policy matches a sentence into one of four possible access control patterns (Table 6). If such a match can be made, Text2Policy classifies the sentence as an ACR.

Once Text2Policy classifies sentences as ACRs, Text2Policy uses the annotated portions of the sentences to extract the subject, action, and object from the sentence. Text2Policy uses a pre-defined domain dictionary to associate the action with specific semantic classes such as "UPDATE" and "DELETE" to determine appropriate permissions for the ACR.

While ACRE and Text2Policy both target the problem of ACR extraction from NL, they differ in their basic approaches. Fundamentally, the differences between ACRE and Text2Policy can be summarized by an inductive versus a deductive approach. ACRE applies inductive reasoning to find and extract ACRs. Text2Policy applies deductive reasoning based upon existing rules (sentence patterns) to find and extract ACRs.

ACRE identifies sentences containing ACRs through a supervised learning approach. As such, the approach requires a labelled dataset similar in structure and content to the document being analyzed. Text2Policy identifies sentences containing ACRs based upon whether or not the shallow parser can parse the sentence into one of its four required patterns. In this regard, Text2Policy has an advantage since Text2Policy does not require a labelled data set to train a classifier. However, Text2Policy can miss ACRs that do not follow one of its four patterns. In our analysis of the documents, we found that only 34.4% of the identified ACR sentences followed one of Text2Policy's patterns. Additionally, Text2Policy's NL parser required splitting longer sentences as the parser could not handle complicated sentence structures.

Text2Policy can extract only one ACR per sentence. For example, ACRE would extract two rules from S4 in Figure 2 (*instructor, enter*$_{create}$*, grade*) and (*instructor, enter*$_{read}$*, student*). Text2Policy would find only the former ACR. From our evaluation in Section 5.3, we found that sentences containing multiple ACRs account for 33% of the examined sentences.

Table 6. Text2Policy - Semantic Role Patterns in Access Control Sentences [35]

Semantic Pattern	Examples
Model Verb in Main Verb Group	An LHCP$_{[subject]}$ **can view**$_{[action]}$ the patient's account$_{[resource]}$. An admin$_{[subject]}$ **should not update**$_{[action]}$ the patient's account$_{[resource]}$.
Passive Voice followed by To-infinitive Phrase	An LHCP$_{[subject]}$ **is disallowed to update**$_{[action]}$ the patient's account$_{[resource]}$. An LHCP$_{[subject]}$ **is allowed to view**$_{[action]}$ the patient's account$_{[resource]}$.
Access Expression	An LHCP$_{[subject]}$ **has read**$_{[action]}$ access to the patient's account$_{[resource]}$. A patient's account$_{[resource]}$ **is accessible**$_{[action]}$ to an LHCP$_{[subject]}$.
Ability Expression	An LHCP$_{[subject]}$ **is able to read**$_{[action]}$ patient's account$_{[resource]}$. An LHCP$_{[subject]}$ **has the ability to read**$_{[action]}$ access to the patient's account$_{[resource]}$.

8. Conclusion

In this paper, we have presented an approach, ACRE, to assist developers in automatically extracting ACRs from NL documents. ACRE provides a way for developers to quickly generate an initial set of ACRs with traceability back to the originating sentences. Developers can apply the approach to detect conflicts in generated rules as well as evaluating the coverage of the generated rules to the identified subjects and resources. We demonstrated how effectively a bootstrapping algorithm can extract rules from a very small initial set of patterns.

We found that ACRs exist in 47% of the examined sentences. Our ACRE approach correctly identifies ACR sentences with a precision of 81% and recall of 65%. The approach extracts ACRs from those identified ACR sentences with an average precision of 76% and an average recall of 49%. Due to the bootstrapping mechanism, the approach works better with a longer document with similar sentences structures, subjects, and resources repeated throughout the document.

9. ACKNOWLEDGMENTS

This work is supported by the USA National Security Agency (NSA) Science of Security Lablet. Any opinions expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSA. We would like to thank the NCSU Realsearch group for their helpful comments on the paper.

10. REFERENCES

[1] 2011 CWE/SANS Top 25 Most Dangerous Software Errors: 2011. *http://cwe.mitre.org/top25/*. Accessed: 2011-11-14.

[2] Brodie, C. a. et al. 2006. An Empirical Study of Natural Language Parsing of Privacy Policy Rules Using the SPARCLE Policy Workbench. *In Proc. SOUPS*. (2006), 8–19.

[3] Carreras, X. and Màrquez, L. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. *In Proc. CoNLL* (2005), 152–164.

[4] Chinchor, N. and Sundheim, B. 1996. Message Understanding Conference - 6: A Brief History. *In Proc. Coling* (1996), 466–471.

[5] Chomsky, N. 1956. Three models for the description of language. *Information Theory, IRE Transactions on*. 2, 3 (1956), 113–124.

[6] Fernandez, E.B. et al. 1997. Determining Role Rights from Use Cases. *In Proc. ACM Workshop on RBAC* (1997), 121 – 125.

[7] Fleiss, J.L. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*. 76, 5 (1971), 378–382.

[8] Fontaine, P.J. 2001. *Goal-Oriented Elaboration of Security Requirements*. Université catholique de Louvain.

[9] Gesmundo, A. and Samardžić, T. 2012. Lemmatisation as a Tagging Task. *In Proce. ACL*. (2012), 368–372.

[10] Han, J. et al. 2011. *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

[11] He, Q. and Antón, A.I. 2009. Requirements-based Access Control Analysis and Policy Specification (ReCAPS). *Information and Software Technology*. 51, 6 (Jun. 2009), 993–1009.

[12] Huddleston, R. and Pullman, G. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press.

[13] IBM 2004. *Course Registration Requirements*.

[14] Inglesant, P. et al. 2008. Expressions of Expertness: The Virtuous Circle of Natural Language for Access Control Policy Specification. *In Proc. SOUPS* (2008), 77–88.

[15] Jurafsky, D. and Martin, J. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson.

[16] Kennedy, C. and Boguraev, B. 1996. Anaphora for everyone: pronominal anaphora resoluation without a parser. *In Proc. Coling* (1996), 113–118.

[17] Landis, J.R. and Koch, G.G. 1977. The Measurement of Observer Agreement for Categorical Data Data for Categorical of Observer Agreement The Measurement. *Biometrics*. 33, 1 (1977), 159–174.

[18] Language-Independent Named Entity Recognition: 2003. *http://www.cnts.ua.ac.be/conll2003/ner/*.

[19] Levenshtein, V.I. 1966. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics Doklady*. 10, 8 (1966), 707–710.

[20] Manning, C. et al. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[21] Manning, C. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? *In Proc. CICLing* (2011), 171–189.

[22] De Marneffe, M.-C. et al. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. *In Proc. LREC* (2006), 449–454.

[23] Meneely, A. et al. 2011. iTrust Electronic Health Care System: A Case Study. *Software System Traceability*.

[24] Piskorski, J. and Yangarber, R. 2013. Information Extraction: Past, Present, and Future. *Multi-source, Multilingual Information Extraction and Summarization*. T. Poibeau et al., eds. Springer Berlin Heidelberg. 23–50.

[25] Potts, C. and Recasens, M. 2012. The Life and Death of Discourse Entities : Identifying Singleton Mentions. 0, Table 1 (2012).

[26] Samarati, P. and Vimercati, S. de 2001. Access control: Policies, models, and mechanisms. *Foundations of Security Analysis and Design*. (2001), 137–196.

[27] Santorini, B. 1995. Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision, 2nd printing). June 1990 (1995).

[28] Schwitter, R. 2010. Controlled Natural Languages for Knowledge Representation. *In Proc. CICLing* (2010), 1113–1121.

[29] Shi, L. and Chadwick, D. 2011. A Controlled Natural Language Interface for Authoring Access Control Policies. *In Proc. SAC* (2011), 1524–1530.

[30] Slankas, J. and Williams, L. 2013. Access Control Policy Extraction from Unconstrained Natural Language Text. *In Proc. PASSAT* (2013), 435–440.

[31] Socher, R. et al. 2013. Parsing with Compositional Vector Grammars. *In Proc. ACL*. (2013).

[32] Stadt, R. Van De 2012. Cyberchair: A web-based groupware application to facilitate the paper reviewing process. *arXiv arXiv:1206.1833*. (2012).

[33] Surdeanu, M. and Johansson, R. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. *In Proc. CoNLL* (2008), 159–177.

[34] Vidya Sagar, V.B.R. and Abirami, S. 2014. Conceptual modeling of natural language functional requirements. *Journal of Systems and Software*. 88, (Feb. 2014), 25–41.

[35] Xiao, X. et al. 2012. Automated Extraction of Security Policies from Natural-Language Software Documents. *In Proc. FSE* (2012).

[36] Zhu, X. et al. 2013. Detecting concept relations in clinical text: insights from a state-of-the-art model. *Journal of biomedical informatics*. 46, 2 (May 2013), 275–85.

[37] Zowghi, D. and Gervasi, V. 2003. On the interplay between consistency, completeness, and correctness in requirements evolution. *Information and Software Technology*. 45, 14 (Nov. 2003), 993–1009.