Towards a Top-down Policy Engineering Framework for Attribute-based Access Control

Masoud Narouei
INSPIRE Lab
University of North Texas
Denton, Texas
Masoudnarouei@my.unt.edu

Hamed Khanpour
Machine Learning Lab
University of North Texas
Denton, Texas
Hamedkhanpour@my.unt.edu

Hassan Takabi
INSPIRE Lab
University of North Texas
Denton, Texas
Hassan.Takabi@unt.edu

Natalie Parde
HiLT Lab
University of North Texas
Denton, Texas
Natalieparde@my.unt.edu

Rodney Nielsen
HiLT Lab
University of North Texas
Denton, Texas
Rodney.Nielsen@unt.edu

ABSTRACT

Attribute-based access control (ABAC) is a logical access control methodology where authorization to perform a set of operations is based on attributes of the user, the objects being accessed, the environment, and a number of other attribute sources that may be relevant to the current request. Once fully implemented within an enterprise, ABAC promotes information sharing while maintaining control of the information. However, the cost of developing ABAC policies can be a significant obstacle for organizations to migrate from traditional access control models to ABAC. Most organizations have high-level requirement specifications that define security policies and include a set of access control policies. Taking advantage of this rich source of information, we introduce a top-down policy engineering framework for ABAC that aims to automatically extract policies from unrestricted natural language documents and then, we present our methodology to extract policy related information using deep neural networks. We first create an annotated dataset comprised of 2660 sentences from real-world policy documents. We then train a deep recurrent neural network (RNN) to identify sentences containing access control policies (ACP) from irrelevant content. We applied the RNN to our new dataset as well as to five other, smaller datasets that have been employed in prior work on this task, and show that our model outperforms the state-of-the-art and leads to a performance improvement of 5.58% over the previously reported results.

CCS CONCEPTS

•Security and privacy → Security requirements; Access control;

KEYWORDS

Access control policy; attribute-based access control; policy engineering; recurrent neural network; deep learning

ACM Reference format:
Masoud Narouei, Hamed Khanpour, Hassan Takabi, Natalie Parde, and Rodney Nielsen. 2017. Towards a Top-down Policy Engineering Framework for Attribute-based Access Control. In *Proceedings of SACMAT'17, June 21–23, 2017, Indianapolis, IN, USA, ,* 12 pages.
DOI: http://dx.doi.org/10.1145/3078861.3078874

SACMAT'17, June 21–23, 2017, Indianapolis, IN, USA
 ACM ISBN 978-1-4503-4702-0/17/06…$15.00.
DOI: http://dx.doi.org/10.1145/3078861.3078874

1 INTRODUCTION

Traditionally, access control has been based on the identity of a user requesting to perform an operation (e.g., write) on a resource (e.g., a database file), either directly, or via predetermined attribute types such as roles or groups assigned to that user. However, this approach has been shown to be hard to manage given the need to associate capabilities directly to users or their roles or groups (e.g. role explosion issue). It has also been noted that using identifiers such as roles and groups are often insufficient in the expression of real-world access control policies (ACPs). To overcome these shortcomings, there has been a growing demand from both government and industry for a more general and dynamic model of access control. This model, which grants or denies a request based on attributes of the user, the objects being accessed, the environment and a number of other attribute sources that may be relevant to the current request, is often referred to as attribute-based access control (ABAC). Previous literature has shown that ABAC is able to overcome the limitations of the dominant access control models (i.e, discretionary access control (DAC), mandatory access control (MAC, also known as lattice based access control or multilevel security), and role-based access control (RBAC)) while unifying their advantages [19].

Using ABAC, flexible enforcement of ACPs can be achieved solely based on the results of a boolean statement comparing different attributes. For example, *"user.type==student AND object.classification ==restricted"* grants permission only based on a subset of user and object attributes which already exist in the system and do not need to be manually entered by administration. As a result of this flexibility, ABAC has attracted interest across industry and government.

In fact, Gartner recently predicted that "by 2020, 70% of enterprises will use attribute-based access control ... as the dominant mechanism to protect critical assets" [11]. However, manual development of initial ABAC policies can be difficult, expensive, labor-intensive, and error prone [3, 16].

ABAC policy mining algorithms have been introduced to reduce the cost of ABAC policy development, by partially automating the process [50]. However, these approaches ignore an important source of policy information in organizations that could be very useful in the policy engineering process. Most organizations have high-level requirement specifications that determine who, under what circumstances, may access what information [16]. These documents define security policies and include a set of ACPs which describes allowable operations for the system. We refer to these documents (high-level requirement specifications) as natural language access control policies (NLACPs) which are defined as "statements governing management and access of enterprise objects. NLACPs are human expressions that can be translated to machine-enforceable access control policies" [16]. We propose to utilize this rich source of information in the process of developing ABAC policies. However, an issue in developing ABAC policies is that the information that needs to be encoded is typically buried within these NLACPs, and difficult to interpret. This requires processing natural language documents and extracting the related information from those documents.

In this paper, we take the first step towards our eventual goal of developing ABAC policies from unrestricted natural language documents (e.g., requirement documents, policy documents, etc.). We introduce a top-down policy engineering framework for ABAC that aims to automatically extract policies from NLACPs and then, we present our methodology to process unrestricted natural language documents and extract policy related information using deep neural networks.

NLACPs are often huge and consist of a lot of general descriptive sentences that lack any access control content. Manually processing these documents to extract policy related information and then using them to build an ABAC policy is a laborious and expensive process. Recent developments in deep learning has surprised many researchers due to high performances in many tasks such as Natural Language Processing (NLP) [41], even leading Manning (2016) to refer to the phenomenon as a neural network "tsunami". The most significant benefit of using deep neural networks is that they are not reliant on handcrafted features; instead, they manufacture features automatically from each word [46], sentence [26], or even long texts [38]. By taking aspirations from the previous reports in effectiveness of deep neural networks in domain-independent conversations [26], we propose a model based on a recurrent neural network, long short term memory (LSTM), that benefits from deep layers of networks and pre-trained word embeddings derived from Wikipedia articles. Word embeddings are semantic distributional representations of words that are used to solve the data sparsity problem [4]. This model will be used to distinguish ACP sentences from non-ACP content. Once the ACP sentences are identified, they can be analyzed in order to extract ABAC policy elements using different methods such as semantic role labeling [34].

The ACP domain suffers from a scarcity of publicly available data for researchers. To help alleviate this problem, we begin by creating a new dataset of real world policy information 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 interested in this same problem to evaluate their own work both more comprehensively and in direct comparison to ours.

The contributions of this paper are hence three-fold:

- We introduce a top-down policy engineering framework for ABAC.
- We take the first step towards developing the framework by proposing an automatic approach to identify access control policy sentences.
- We create a deep recurrent neural network that uses pre-trained word embeddings to identify sentences that contain access control policy content and show that the method is effective in doing so.

The rest of this paper is organized as follows: We begin with an overview of background information in section 2. This is followed by introduction of the proposed top-down policy engineering framework and its components in section 3. Section 4 presents the proposed ACP sentence identification and deep learning based approach in detail. The experiments and results are presented in section 5, followed by review of literature in section 6. Finally, the conclusion and future work wraps up the paper.

2 BACKGROUND

This section provides background information with regards to semantic role labeling and deep learning.

2.1 Semantic Role Labeling

Semantic role labeling (SRL), sometimes also called shallow semantic parsing, is a task in NLP for automatically identifying the semantic roles of each argument of each predicate (verb) in a sentence [12]. With the advent of resources such as FrameNet [9] and PropBank [36], SRL has experienced a flurry of activity in recent years [6]. SRL labels verb-argument structures using the notation defined by these resources, identifying who did what to whom where and when by assigning roles to constituents of the sentence representing entities related to a specific predicate. Answering these questions is the key step in automatic conversion of ACP sentences to ABAC elements (subject, operation, object). The following example represents the annotation of semantic roles using SRL:

[A_0 The system] [V **retrieves**] [A_1 the student information] [$AM - LOC$ in the registration system].

Here, the roles for the predicate retrieves (retrieves.01, that is, the roleset of the predicate) are defined in the PropBank Frames scheme as:

- **Arg0-PAG**: *receiver (vnrole: 13.5.2-agent)*
- **Arg1-PPT**: *thing gotten (vnrole: 13.5.2-theme)*
- **Arg2-DIR**: *received from (vnrole: 13.5.2-source)*
- **Arg3-GOL**: *benefactive*

The Proposition Bank [36], generally referred to as PropBank, is a resource of sentences annotated with semantic roles. Since it is difficult to define a universal set of presentive roles, the semantic roles in PropBank are defined with respect to individual verb senses. Each sense, therefore, has a specific set of roles, which are given

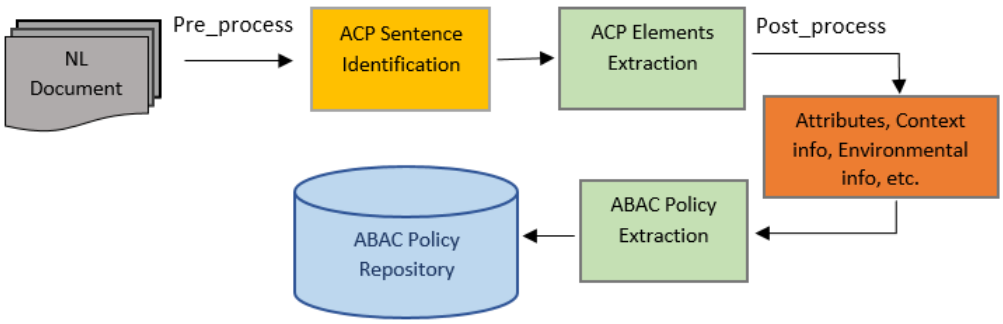

Figure 1: Overview of the Proposed Top-down Policy Engineering Framework

only numbers rather than names: Arg0, Arg1, Arg2, and so on. In general, Arg0 represents the PROTO-AGENT (subject), and Arg1, the PROTO-PATIENT (object or resource). The semantics of the other roles are less consistent, often being defined specifically for each verb. Nonetheless, there are some generalizations; the Arg2 is often the benefactive, instrument, attribute, or end state, the Arg3 the start point, benefactive, instrument, or attribute, and the Arg4 the end point [20]. PropBank has also a number of non-numbered arguments called ArgMs (ArgM-TMP, ArgM-LOC, Arg-DIR, etc.), which are relatively stable across predicates. Using these modifiers allow us to detect temporal or location of a specific ACP.

2.2 Deep Learning

Deep learning (also known as deep structured learning, hierarchical learning or deep machine learning) is a learning method that attempts to model high level abstractions in data using a deep network [8]. In a deep network, there are many layers between the input and output, allowing the algorithm to use multiple processing layers, composed of multiple linear and non-linear transformations. The first layer receives an input signal and then transforms it by a processing unit, like an artificial neuron, whose parameters are "learned" through training. Then it passes its transformed output to the next layer. Each successive layer uses the output from the previous layer as input. Deep learning has been used in both supervised or unsupervised settings and its applications include pattern analysis (unsupervised) and classification (supervised).

3 THE PROPOSED TOP-DOWN POLICY ENGINEERING FRAMEWORK

In order to extract ABAC policies from unrestricted natural language documents, we first need to process the documents and identify those sentences that carry ACP content. The ACPs describes who has access to what resource in what way. Once the ACP sentences are identified, our proposed framework will generate corresponding policy elements (e.g. subject, object, action). The framework also extract attributes, context information, environmental conditions, etc. Then, this information is used to generate machine readable and enforceable ABAC policies. An overall view of the the proposed framework is shown in Figure 1. In the following sections, we describe each of these steps in detail.

3.1 ACP Sentence Identification

Often time NLACPs contain contents that describe functional requirements and are not necessary related to ACPs. Although these documents also contain ACPs, attempting to extract ACPs from the whole document is an error prone and tedious process. To correctly extract ACPs from NLACPs, it is very important to find out those sentences that have potentially ACP content and then perform further analysis only on those sentences to extract ACP elements. This step is one of the main contributions of this paper and discussed in more detail in Section 4 and Section 5.

3.2 ABAC Policy Elements Extraction

As we mentioned earlier, an issue prior to developing an ABAC policy is the information that needs to be encoded is typically buried within existing natural language (NL) artifacts, hence difficult to interpret. For example, consider the following ACP sentence for iTrust [31], "System displays only the applicable input entries to the UAP." This ACP sentence is not amenable for automated verification and enforcement, requiring manual effort in extracting the necessary elements (e,g., subject, object, action) from this sentence. To address this issue, various researchers have proposed approaches for automatically generating machine-enforceable ACPs from NL software documents in different formats such as eXtensible access control markup language (XACML) [49]. *ACRE* [40] used an iterative algorithm to discover patterns that represent ACP rules in sentences. They seeded this algorithm with frequently occurring nouns matching a subject-action-resource pattern throughout a document. The algorithm then searched for additional combinations of those nouns to discover additional patterns. The found instances were assumed to represent ACPs and the elements of the ACP were then extracted. In our previous work, we proposed SRL to automatically extract ACP elements from unrestricted NL documents, define roles, and build an RBAC model [34]. We did not attempt to identify ACP sentences, but instead used the already extracted sentences by [40] and left implementing the ACP sentence identification step for future work. By applying SRL on the ACP

sentences to automatically identify predicate-argument structures, and a set of predefined rules on the extracted arguments, we were able to correctly identify ACPs with a precision of 75%, recall of 88%, and F_1 score of 80%. On average, our method bested *ACRE* with 2% increase in F_1. In this work, we extend that work and propose our methodology for ACP sentence identification alongside a new framework for ABAC policy engineering. Since our goal is to automate the process of generating ABAC policies, we adopt the same process as [34] and use the SRL arguments, extracted from analyzing each sentence, as our basic ABAC components.

3.3 Attribute Extraction

One of the main challenges in developing an ABAC policy is identifying attributes (e.g, subject attribute, object attributes and environment attributes). Most of the recent works assume that these attributes are provided as part of the data [50]; however, in real scenarios, especially while analyzing policy documents, no attributes are provided. To tackle this issue, we propose using SRL argument definitions as descriptors. In Propbank [36], in addition to the arguments for each predicate, a short definition is also provided for each argument. Considering the following roleset for *retrieve* in the sectence "The system retrieves the student information in the registration system":

- **Arg0-PAG**: *receiver (vnrole: 13.5.2-agent)*
- **Arg1-PPT**: *thing gotten (vnrole: 13.5.2-theme)*
- **Arg2-DIR**: *received from (vnrole: 13.5.2-source)*
- **Arg3-GOL**: *benefactive*

The argument A_0 has a short definition of *receiver*, and the argument A_1 has the definition of *thing gotten*, etc. Note that the definitions in the frame file for each role ("thing gotten", "received from") are informal glosses intended to be read by humans, rather than being formal definitions. However, these definitions are rather sufficient for our purpose. Hence, we consider *receiver* as the descriptor for A_0 and *thing gotten* as the descriptor for A_1. For our basic prototype, we only consider A_0 and A_1 and leave the rest of numbered attributes for future work. We use A_0's descriptor as subject attribute and A_1's as object attribute. Since some ACP predicates also include non-numbered arguments of *ArgM-TMP* and *ArgM-LOC*, we consider them as environment attributes. We consider *ArgM-TMP* as temporal attribute and *ArgM-LOC* as location attribute.

Following the above descriptions, we use the following function notation for the value assignment of attributes:

$$Definition(Argument) = Text \tag{1}$$

As an example, consider the following sentence:

[A_0 The system] [V **retrieves**] [A_1 the student information] [*ArgM-LOC* in the registration system].

Using the above function notation, the attributes would be represented as follows:

receiver(A_0)= "The system"
thing-gotten(A_1)= "the student information"
Location(*ArgM-LOC*)= "in the registration system"

We define *Sbj_Att* as the AND combination of subject attributes, *Obj_Att* as the AND combination of object attributes and *Env_Att* as the AND combination of environment attributes. In its most general form, an ABAC policy is a combination of individual attributes. We represent an ABAC policy using the following notation:

$$Policy : Action \leftarrow \{Sbj\_Att \wedge Obj\_Att \wedge Env\_Att\} \tag{2}$$

Using this notation, the above policy would be represented as the following:

retrieve $\leftarrow$ {
receiver(A_0) = "The system" $\wedge$
thing-gotten(A_1)= "the student information" $\wedge$
Location(*ArgM-LOC*)= "in the registration system" }

This process is further depicted in Figure 2 using another ACP example.

4 THE PROPOSED ACCESS CONTROL POLICY SENTENCE IDENTIFICATION

In this section, we describe the ACP sentence identifier component which is the first step in top-down policy engineering process. The ACP sentence identification consists of a pre-process engine, and RNN sentence classifier as described below.

4.1 Pre-process Engine

Figure 3 presents part of a large policy document. It is obvious that there are many non-relevant contents such as titles, tables, etc. that need to be removed. As these formal 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 order to parse the extracted text, we feed it into Stanford CoreNLP toolkit [30]. The tool will split the text by sentences boundaries where each sentence will be 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.

$$Ratio(sent) = \frac{Capitals(sent)}{Tokens(sent)} \tag{3}$$

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 will be extracted from the document [2]:

- *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*

[1] https://pdfbox.apache.org/index.html
[2] http://policy.unt.edu/sites/default/files/07.022_AdministrativeEntrySearchesUniversityResidenceHalls_201.pdf

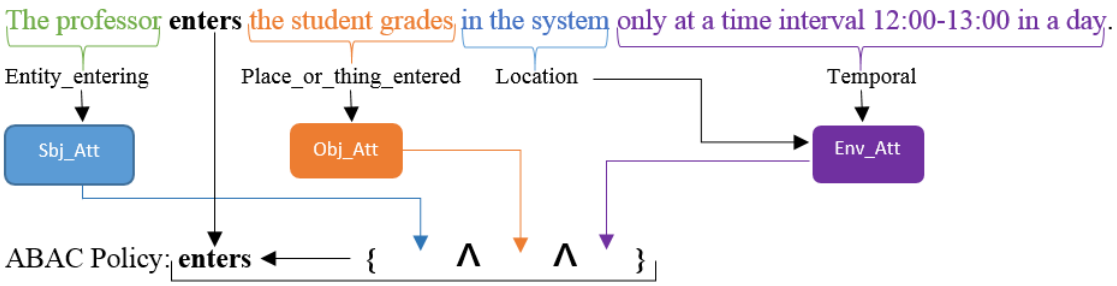

Figure 2: Attribute Extraction Scenario

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.

Figure 3: Part of a requirements document.

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 will be fed to the deep neural network classifier in order to build the network and make predictions.

4.2 Recurrent Neural Network (RNN) Sentence Classifier

Recently, DNNs have been used with increasing frequency in a variety of text processing applications, from sentiment analysis [41] to conversational text processing for dialogue systems [22, 48]. Collobert *et al.* proposed an approach for generating word vectors based on contextual information gained from large amounts of unlabeled text, employing convolutional neural networks (CNNs) to develop an efficient application for part-of-speech tagging, chunking, named entity recognition, semantic role labeling, and syntactic parsing, called *SENNA* [7]. They showed that their developed system outperformed almost all sophisticated traditional methods that perform these same NLP tasks, with the substantial added benefit of not needing to employ any handcrafted features, prior knowledge, or linguistic information. In this work, we will create a deep recurrent neural network (RNN) that uses pre-trained word embeddings in order to identify sentences that contain ACPs from large requirements documents.

Current approaches based on deep learning methods improved many state-of-the-art techniques in NLP, including dialogue act (DA) classification accuracy on open-domain conversations [26, 39]. Kalchbrenner and Blunsom used a mixture of CNN and RNN [21]. CNNs were used to extract local features from each utterance, and RNNs were used to create a general view of the whole dialogue. This work improved the state-of-the-art 42-tag DA classification on Switchboard [42] by 2.9% to reach 73.9% accuracy. Recently, Ji *et al.* presented a hybrid architecture that merges an RNN language model with a discourse structure that considers relations between two contiguous utterances as a latent variable [26]. This approach improved the result of the state-of-the-art method by about 3% (from 73.9 to 77) when applied on the Switchboard corpus.

4.2.1 Text Representation by Recurrent Neural Network. Figure 4 depicts a general view of an RNN that makes use of sequential information by building connections between current and previous inputs. This specificity is particularly meaningful when processing sequences of text in which each word is syntactically and semantically linked with previous words.

As can be seen in Figure 4, information from previous states is provided to the current state t by the previous hidden layer (h_{t-1}), which contributes to building h_t. We chose to use the RNN model to identify ACP sentences because it was designed to consider units in sequence to create a deterministic probability distribution over hidden layers, which enables the model to preserve much more information than its counterparts (e.g., HMM and CRF). Since in our task we need to discover specific relations among words in the sentence, our model needs to extract syntactic and semantic features that relate to one another sequentially. This makes RNNs the most appropriate choice since they are best able to contribute and remember the extracted features from each step.

The inputs of our model are word embeddings; each word from the sentence is represented as a vector, and each sentence is a vector

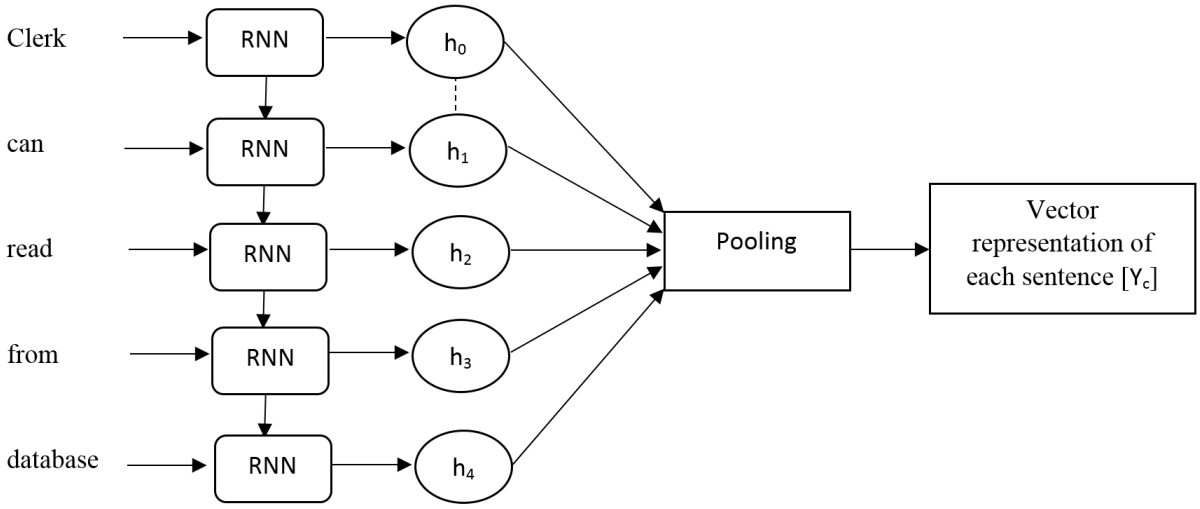

Figure 4: RNN structure for creating a vector-based representation of a sentence from its word.

of these word vectors, $< T_1, T_2, ..., T_{t-1}, T_t, ..., T_n >$. h_t is defined as follows:

$$h_t = \sigma\left(W^{hh}h_{t-1} + W^{hd}T_t\right) \tag{4}$$

where $W^{hh}\varepsilon R^{h\times h}$ and $W^{hd}\varepsilon R^{h\times d}$ are weight matrices, and σ refers to the logistic sigmoid function. Further, $y_t\varepsilon R^K$ is the class representation of each sentence, with K representing the number of classes for classification task (e.g., ACP and non-ACP). y_t is defined as follows:

$$y_t = softmax\left(W^{(S)}h_t\right) \tag{5}$$

In the pooling layer of our RNN (see Figure 4), our model takes all h vectors, $h_{1:n}$, and generates one output vector that carries the best features, namely those that are the best fit for the sentence. Performing this action can be done in one of three ways: mean-, max- or last-pooling. Mean-pooling measures the average of all h vectors, max-pooling takes the maximums from each h vector, and last-pooling takes the last h vector (i.e., h_t).

Although basic RNNs are theoretically able to carry the history of the network, they fail to maintain information over long distances in the text [5, 15]. Another problem with basic RNNs lies in the development of vanishing and exploding gradients, which lead the network to terminate prematurely [33, 37].

Long Short term memory (LSTM) networks are a variation of the RNN structure that are able to solve some of these issues. The LSTM structure is adjusted so that it holds long-distance relations as its default specificity. Using LSTMs results in a greater level of assurance that any useful features in the text will be captured and preserved. Since preserving trivial features, on the other hand, can be harmful to the classifier's result, a *forget gate layer* is designed in LSTMs to discard trivial low weight features from the cell state (see Eq. 7). Figure 5 shows the standard structure of an LSTM cell.

As can be seen in Figure 5, the LSTM cell at each time step t is defined by a set of vectors in R^d:

$$i_t = \sigma\left(W^{(i)}T_t + U^{(i)}h_{t-1} + b^{(i)}\right) \tag{6}$$

$$f_t = \sigma\left(W^{(f)}T_t + U^{(f)}h_{t-1} + b^{(f)}\right) \tag{7}$$

$$o_t = \sigma\left(W^{(o)}T_t + U^{(o)}h_{t-1} + b^{(o)}\right) \tag{8}$$

$$u_t = \tanh\left(W^{(u)}T_t + U^{(u)}h_{t-1} + b^{(u)}\right) \tag{9}$$

$$C_t = i_t \bigodot u_t + f_{(t)} \bigodot c_{t-1} \tag{10}$$

$$h_t = o_t \bigodot \tanh(c_t) \tag{11}$$

where i_t is the input gate, f_t is the forget gate, o_t is the output gate, c_t is the memory cell, h_t is the hidden state, and $\bigodot$ represents element-wise multiplication.

LSTMs have gates in each cell that control the types of signals that are allowed to pass through the whole chain. For instance, the forget gate f_t (Eq. 7) controls the amount with which the previous memory should be forgotten, the input gate (Eq. 6) controls the updating process, and the output gate controls the extent to which the internal memory state should be changed. The hidden layer h_t represents a gated, partial view of its cell state. LSTMs are able to view information over multiple time scales due to the fact that gating variables are assigned different values for each vector element [45].

By setting LSTM cells back to back, each cell is connected to one another via the hidden layer, h_t [13, 43]. This configuration of stacked LSTM cells is used to facilitate the identification of dependencies between tokens (T) across longer distances in the input chain of words.

In this study, stacked LSTMs with pre-trained word embeddings are used. We trained our word embeddings on Wikipedia using 300-dimensional vectors, setting our window and min-count parameters equal to 5 [32].

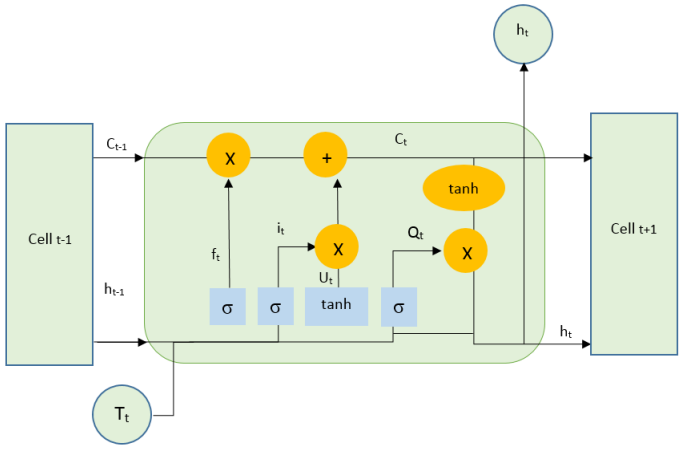

Figure 5: LSTM cell structure and parameters (http://colah.github.io).

5 EXPERIMENTS AND EVALUATION

5.1 Dataset

As mentioned earlier, the access policy domain suffers from a scarcity of publicly available data for researchers. To help alleviate this problem, 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 interested in this same problem to evaluate their own 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 authorizations for a wide variety of departments, including Human Resources, 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, we limited our data to a randomly-selected subset of 2,660 sentences from the pool of sentences.

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 first author of this paper adjudicated any discrepancies in the annotations after discussing them with both annotators. We computed Fleiss' kappa [10] on the annotations, finding $\kappa = 0.75$ between the two annotators. According to guidelines provided by Landis' and Koch's, scores between 0.41 and 0.60 indicate moderate agreement, scores between 0.61 and 0.80 indicate substantial agreement, and scores between 0.81 to 1.00 are considered almost perfect agreement [25].

The final annotated dataset is comprised of 1,460 ACP sentences and 1,200 non-ACP sentences. This dataset is available upon request.

[3]https://policy.unt.edu/
[4]https://app.unthsc.edu/policies

In addition to evaluating our approach on this new dataset, we also evaluated it on the same data that has been used in previous research in this area [40], to provide a direct comparison with the state-of-the-art. These prior datasets were manually labeled by Slankas *et al.* [40], and are described as follows:

- **iTrust for ACRE:** *iTrust* is an open source healthcare application for which 40 use-cases plus additional non-functional requirements are available. Two different versions of the *iTrust* data exist; this version, *iTrust for ACRE*, was extracted directly from the project's wiki [5] [31].
- **iTrust for Text2policy:** This second version of the *iTrust* data was taken from the documentations used by Xiao *et al.* [49].
- **IBM Course Management:** This dataset is comprised of eight use-cases from the IBM Course Registration System [17].
- **CyberChair:** This dataset is comprised of the *CyberChair* documents, which have been used in a variety of contexts across 475 different conferences and workshops [47].
- **Collected ACP Documents:** This dataset is comprised of a combined document of 142 sentences collected by Xiao *et al.* [49].

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 non-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

[5]http://agile.csc.ncsu.edu/iTrust/wiki/doku.php

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.

5.3 Experimental settings

We used our new dataset, referred to herein as ACPData, to tune all necessary parameters such as the word embedding resource type, the word vectors' length, the decay rate, and the number of LSTM cells to be included in the network. We split the dataset into separate training, development, and test sets. The training process was stopped if the resultant F_1 value did not change for 20 consecutive epochs. We set one hyper-parameter value at a time and obtained the F_1 on the development set. We used the following parameter settings as our default settings and changed them one at a time to tune the performance on the development set. These settings are: *drop-out = 0.5, decay rate = 0.5, layer size = 2*, and Word2Vec (Wikipedia) with 300 dimensions.

5.4 Word Embedding Settings

We conducted experiments to determine the best word embedding settings (i.e., word embedding method, vector length, and resource type) for our data. We used pre-trained word vectors provided by Mikolov *et al.* [32] and Pennington *et al.* [38], which were trained on the Google News and Common Crawl datasets using the Word2Vec [32] and Glove [38] methods, respectively. We also generated 300-dimensional word vectors using the Word2Vec package [32], with Wikipedia as our resource. Table 1 shows the results obtained when applying different word-embedding parameters.

F_1 (%)	Resource	Dimension
83.0	Word2vec (Wikipedia)	75
83.5	Word2vec (Wikipedia)	150
84.91	Word2vec (Wikipedia)	300
86.0	Word2vec (GoogleNews)	75
86.6	**Word2vec (GoogleNews)**	**150**
86.3	Word2vec (GoogleNews)	300
83.5	Glove	75
85.1	Glove	150
82.7	Glove	300

Table 1: Comparison of F_1 using different word embedding parameters.

As can be seen in Table 1, our model achieved its best results on the development set using the word embeddings trained on Google News with 150 dimensions; thus, we use this setting in the remainder of our experiments.

5.5 Decay Rate

Equation 12 describes how the network's connection weights are adjusted, where E represents the error and W_{ij} represents the weight matrix between two nodes, i and j.

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}, \tag{12}$$

To avoid overfitting, a regularization factor was added to Equation 12 to penalize large changes in w_{ij}, thus leading to the updated Equation 13 below.

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} - \eta \lambda w_{ij}. \tag{13}$$

The term $-\eta\lambda w_{ij}$ is the regularization factor, with η as the learning rate, and λ as the decay factor that causes w_{ij} to decay in scale to its prior measure.

We tuned these parameters on our development set, comparing the resultant accuracies when using different decay rates (analysis shown in Table 2). As can be seen, the accuracy of the system peaks when λ is set to 0.7. Therefore, we set $\lambda = 0.7$ in our final experiments.

F_1 (%)	λ
84.59	0.1
84.63	0.2
84.68	0.3
84.76	0.4
84.91	0.5
85.17	0.6
85.53	0.7
84.42	0.8

Table 2: Decay factor λ vs. F_1.

5.6 Dropout

Dropout regularization is one of the most common methods for regularization among approaches that incorporate deep learning [14], and is designed to diminish the network's chances of overfitting. It works by "dropping out" some of the nodes in each training iteration, by randomly assigning a value of zero to their weights. The underlying goal in doing this is to encourage the nodes in the network to become less dependent on the other nodes to which they are connected. Dropout methods were originally introduced for feed-forward and convolutional neural networks, but recently have also been applied to the input embeddings layer of some recurrent networks, including LSTMs [2, 35, 51].

Bayer *et al.* showed that standard dropout does not work effectively with RNNs due to noise magnification in the recurrent process, which consequently results in malfunctions in the learning process [2] . Therefore, instead of using standard dropout, we apply the dropout technique proposed by Zaremba *et al.* [51] for regularizing RNNs, which is used by most studies that employ LSTM models [18, 27–29, 44]. Zaremba *et al.* [51] postulate that

their approach reduces overfitting on a variety of tasks, including language modeling, speech recognition, image caption generation, and machine translation. We analyzed the performance of dropout probability measures ranging from 0.0 to 0.6 on our development set to learn the best dropout probability measure to apply in our final experiments.

As is shown in Table 3, we observed that by decreasing the dropout probability measure, our system's performace in terms of F_1 increased continuously. This led us to conclude that including dropout regularization has a negative impact on accuracy for this task; thus, we chose not to use any kind of dropout in our final model.

F_1 (%)	Dropout probability
84.11	0.6
84.91	0.5
84.53	0.4
84.87	0.3
84.00	0.2
84.12	0.1
84.14	0.0

Table 3: Dropout vs. F_1.

5.7 Number of Layers

After tuning the decay rate and dropout probability, we empirically determined the number of LSTM cells to include in our model. This measure influenced the extent to which the model was able to detect relevancy between two tokens within a text (e.g., a model with only one LSTM cell would have been unable to detect relevant tokens that were distant from one another, whereas a model with too many LSTM cells would have been prone to overfitting). Table 4 illustrates the impacts of including varying numbers of LSTM cells on our system's performance on the development set.

F_1 (%)	Depth
84.37	1
84.91	2
85.68	3
85.81	4
85.98	5
85.87	6
84.7	7
83.70	8
83.12	9
82.74	10

Table 4: Number of stacked LSTM cells vs. accuracy.

In addition to tuning the aforementioned parameters, we ran several experiments to learn the impacts of varying *L2-reg*, *pooling*, *activation, and SGD type*, and ultimately set those values as $1e-5$, *last pooling*, *tanh*, and *ADAM* [23], respectively. These settings were consistent with previous findings in the literature, and we did not observe significant improvements by altering these values.

5.8 Experimental Results

The settings described in the previous section were tuned using our ACPData development set. After determining the optimal parameters for identifying access policy sentences using this set, we then applied the resulting settings when evaluating the overall performance of the system using all six of the datasets described earlier (ours plus five others).

Our dataset (ACPData) includes separate training and test sets. For the smaller, pre-existing datasets, we considered each as a separate fold of data, and trained on four of the datasets while testing on the fifth (i.e., *document-fold validation*). For instance, when evaluating our model on the *iTrust for Text2Policy* dataset, we combined the *IBM Course Management*, *CyberChair*, *iTrust for ACRE*, and *Collected ACP* datasets to create a single training set. This configuration allowed us to present a fair comparison between our model and the results presented in Slankas *et al.* [40] (referred to herein as *ACRE*), which were also produced using *document-fold validation* with these datasets. In addition to ensuring a fair comparison with prior work, analyzing the performance of our model on these datasets allowed us a more comprehensive evaluation of our model on many different ACP domains. We compared the results obtained from applying our model against the results obtained from applying a support vector machine (SVM) trained on a selection of n-gram features optimized for each dataset.

Table 5 presents the individual results for (1) our dataset, (2) each of the test folds used in the document-fold validation (results shown for each of these folds were obtained using our model, not *ACRE*), and (3) the overall document-fold validation results for both *ACRE* and our model. For each of the individual datasets, the results are presented for both our model and the baseline model. As can be seen, our model outperformed the SVM approach in all cases. Furthermore, it also led to a 5.58% improvement over *ACRE*. Worthy of note is the observation that our model consistently achieved higher precision than recall. This may have been a consequence of the training sets having originated from different domains (ranging from health care to conference management), which could have allowed the model to capture a variety of patterns while forgetting unrelated ones via the forget gate (a behavior that contrasts with that of traditional classifiers). Nevertheless, in comparison to *ACRE*, our model was able to extract more comprehensive features from the text, which enabled it to extract more access control sentences while not only maintaining its precision, but actually increasing it a small amount.

5.9 Discussion

In this section, we discuss threats to validity of the proposed approach and how they can be mitigated.

The threats to external validity include lack of representativeness of datasets, SRL tools domain dependence and long-term dependency problem in RNNs. The five datasets used in previous literature covered mostly limited grammars and many of their policies were

Corpus	Precision	Recall	F_1 (Our Model)	F_1 (SVM)
ACPData	89.12	88.91	89.00	70.86
iTrust for Text2Policy	77.19	72.46	74.75	64.10
iTrust for ACRE	92.10	76.20	83.40	81.50
IBM Course Management	86.62	82.24	84.37	80.53
CyberChair	76.25	70.19	73.09	64.59
Collected ACP	74.23	69.97	72.00	34.95

Corpus (Model)	Precision	Recall	F_1
Document-Fold Validation (ACRE)	81.00	65.00	72.00
Document-Fold Validation (Our Model)	**81.28**	**74.21**	**77.58**

Table 5: Individual dataset comparisons between our model, SVM, and a comparison between our model and the method proposed by [40] (ACRE) using leave-one-out cross validation at the dataset level.

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 author's 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. As another external threat, a current issue with SRL tools is that they are not consistent with specific target domains such as ACP domain. This is due to the fact that they were trained on publicly available corpora such as PropBank [36], which was taken from the Wall Street Journal. This means that the predicate-argument frames are usually specific to that domain, in this case, largely financial articles. One future direction would be to explore the idea of improving SRL tool's performance using domain-related knowledge.

As the last external threat, LSTMs are specifically designed to avoid the long-term dependency problem in RNNs. However, in many cases we have some important information from the past which are not equated appropriately. One possible solution to improve current structure of the network is to consider an external memory to preserve important data [1] for longer time.

The threats to internal validity include human factors for determining correct identification of ACP sentences from natural language documents. 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 first author of this paper adjudicated any discrepancies in the annotations after discussing them with both annotators.

6 RELATED WORK

There are notably two previous reports on identifying ACP from non-ACP sentences. These two reports sought to identify sentences containing ACPs through the use of predefined patterns or existing machine learning approaches. 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 [49]. An example pattern is the *Modal Verb in Main Verb* group, which positively identifies sentences as ACP sentences if the main verb contains a modal verb. If a pattern is successfully matched, *Text2Policy* uses the annotated portions of the sentence to extract the subject, action, and object from the sentence. Since *Text2Policy* is rule-based, it does not require a labeled training set. However, this comes at a great cost; it misses any ACP sentence that do not follow one of its four handcrafted patterns. It has been reported that only 34.4 % of identified ACP sentences follow one of *Text2Policy*'s patterns [40].

Slankas *et al.* proposed Access Control Rule Extraction (ACRE), a supervised learning approach that uses an ensemble of classifiers (composed of k-nearest neighbors (k-NN), naïve bayes, and support vector machine classifiers) to determine whether a sentence expresses an Access Control Rule or not [40]. To determine which classifier(s) to use, they defined a threshold value of 0.6 based on the k-NN classifier's computed distance between an instance and its neighbors, and the number of words in the sentence. If the computed distance for a test instance fell below the threshold, they returned the k-NN classifier's prediction. Otherwise, the output label returned was the result of a majority vote among the three classifiers.

Other similar approaches reported high performances in classifying user reviews. Kong *et al.* presented AUTOREB, a system that automatically identifies the security and privacy-related behaviors in Android apps by analyzing user reviews [24]. AUTOREB employed state-of-the-art machine learning techniques to infer the relations between users' reviews and four categories of security-related behaviors, namely spam, financial issues, over-privileged permissions, and data leakage. To do so, they adopted a keyword-based approach. The keywords were manually selected in an iterative fashion. The initial set of keywords included "security" and "privacy," and then new key words were selected from those that had a high co-occurrence with the current set of key words. This process was iterated until no more key words were selected. The system achieved an average accuracy of 94.05% in inferring the security behaviors from user reviews. However, one issue with their

system was in determining the thresholds for feature learning and classification. Determination of these thresholds generally required both domain knowledge and cross validation.

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 introduced a top-down policy engineering framework and presented our methodology to extract ABAC policies from organizations' natural language documents. We empirically determined the optimal parameters for, and subsequently applied, a deep recurrent neural network to the task of identifying sentences containing access policy content in unstructured natural language documents. Moreover, we created an annotated dataset comprised of 2660 sentences from real-world policy documents. We applied our model to this new dataset as well as to five other, smaller datasets that have been employed in prior work, and showed that our model outperformed the SVM model results and led to a performance improvement of 5.58% over the state-of-the-art. Our results provided evidence that RNNs are well-suited to the task of access control policy identification, and that they are able to automatically generate more valuable features than those that were handcrafted in prior reports. We also presented the basic idea of our prototype for extracting ABAC policies from ACP sentences. Future works include a comprehensive presentation of our framework for extracting ABAC policies in order to support features required for real-world use of ABAC systems.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Justin Bayer, Christian Osendorfer, Daniela Korhammer, Nutan Chen, Sebastian Urban, and Patrick van der Smagt. 2013. On fast dropout and its applicability to recurrent networks. *arXiv preprint arXiv:1311.0701* (2013).

[3] Matthias Beckerle and Leonardo A Martucci. 2013. Formal definitions for usable access control rule sets from goals to metrics. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*. ACM, 2.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *journal of machine learning research* 3, Feb (2003), 1137–1155.

[5] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5, 2 (1994), 157–166.

[6] Steven Bethard, Zhiyong Lu, James H Martin, and Lawrence Hunter. 2008. Semantic role labeling for protein transport predicates. *BMC bioinformatics* 9, 1 (2008), 277.

[7] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12 (2011), 2493–2537.

[8] Li Deng, Dong Yu, and others. 2014. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing* 7, 3–4 (2014), 197–387.

[9] Charles J Fillmore, Charles Wooters, and Collin F Baker. 2001. *Building a large lexical databank which provides deep semantics*. Citeseer.

[10] Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76, 5 (1971), 378.

[11] Gartner. 2014. Market Trends: Cloud-Based Security Services Market, Worldwide. (2014). https://www.gartner.com/doc/2607617

[12] Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics* 28, 3 (2002), 245–288.

[13] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 273–278.

[14] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580 (2012).

[15] Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München* (1991), 91.

[16] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, and others. 2013. Guide to attribute based access control (ABAC) definition and considerations (draft). *NIST special publication* 800, 162 (2013).

[17] IBM. 2004. Course Registration Requirements. (2004).

[18] Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain Adaptation of Recurrent Neural Networks for Natural Language Understanding. *arXiv preprint arXiv:1604.00117* (2016).

[19] Xin Jin, Ram Krishnan, and Ravi Sandhu. 2012. A unified attribute-based access control model covering DAC, MAC and RBAC. In *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 41–55.

[20] Dan Jurafsky and James H Martin. 2014. *Speech and language processing*. Vol. 3.

[21] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *arXiv preprint arXiv:1306.3584* (2013).

[22] Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue Act Classification in Domain-Independent Conversations Using a Deep Recurrent Neural Network.. In *COLING*.

[23] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[24] Deguang Kong, Lei Cen, and Hongxia Jin. 2015. Autoreb: Automatically understanding the review-to-behavior fidelity in android applications. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 530–541.

[25] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

[26] Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827* (2016).

[27] Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding CNNs for text: non-linear, non-consecutive convolutions. *arXiv preprint arXiv:1508.04112* (2015).

[28] Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Lluís Màrquez. 2016. Semi-supervised Question Retrieval with Gated Convolutions. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, 1279–1289. http://www.aclweb.org/anthology/N16-1153

[29] Liang Lu, Lingpeng Kong, Chris Dyer, Noah A Smith, and Steve Renals. 2016. Segmental Recurrent Neural Networks for End-to-end Speech Recognition. *arXiv preprint arXiv:1603.00223* (2016).

[30] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit.. In *ACL (System Demonstrations)*. 55–60.

[31] Andrew Meneely, Ben Smith, and Laurie Williams. 2011. iTrust Electronic Health Care System: A Case Study. *Software and Systems Traceability* (2011).

[32] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[33] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model.. In *Interspeech*, Vol. 2. 3.

[34] Masoud Narouei and Hassan Takabi. 2015. 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*. ACM, 157–160.

[35] Marius Pachitariu and Maneesh Sahani. 2013. Regularization and nonlinearities for neural language models: when are they needed? *arXiv preprint arXiv:1301.5650* (2013).

[36] Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31, 1 (2005), 71–106.

[37] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)* 28 (2013), 1310–1318.

[38] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation.. In *EMNLP*, Vol. 14. 1532–1543.

[39] Suman Ravuri and Andreas Stoicke. 2015. A comparative study of neural network models for lexical intent classification. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 368–374.

[40] John Slankas, Xusheng Xiao, Laurie Williams, and Tao Xie. 2014. Relation extraction for inferring access control rules from natural language artifacts. In *Proceedings of the 30th Annual Computer Security Applications Conference*. ACM, 366–375.

[41] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for

semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, Vol. 1631. Citeseer, 1642.

[42] Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics* 26, 3 (2000), 339–373.

[43] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[44] Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Greedy, Joint Syntactic-Semantic Parsing with Stack LSTMs. *arXiv preprint arXiv:1606.08954* (2016).

[45] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).

[46] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, 384–394.

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

[48] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745* (2015).

[49] Xusheng Xiao, Amit Paradkar, Suresh Thummalapenta, and Tao Xie. 2012. 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*. ACM, 12.

[50] Zhongyuan Xu and Scott D Stoller. 2015. Mining attribute-based access control policies. *IEEE Transactions on Dependable and Secure Computing* 12, 5 (2015), 533–545.

[51] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* (2014).