2012 IEEE International Symposium on Policies for Distributed Systems and Networks

Classifying Natural Language Sentences for Policy

John Slankas and Laurie Williams
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
[john.slankas,laurie_williams]@ncsu.edu

***Abstract*—Organizations derive policies from a wide variety of sources, such business plans, laws, regulations, and contracts. However, an efficient process does not yet exist for quickly finding or automatically deriving policies from uncontrolled natural language sources. *The goal of our research is to assure compliance with established policies by ensuring policies in existing natural language texts are discovered, appropriately represented, and implemented.* We propose a tool-based process to parse natural language documents, learn which statements signify policy, and then generate appropriate policy representations. To evaluate the initial work on our process, we analyze four data use agreements for a particular project and classify sentences as to whether or not they pertain to policy, requirements, or neither. Our k-nearest neighbor classifier with a unique distance metric had a precision of 0.82 and a recall of 0.81, outperforming weighted random guess, which had a precision of 0.44 and a recall of 0.46. The initial results demonstrate the feasibility of classifying sentences for policy and we plan to continue this work to derive policy elements from the natural language text.**

Keywords-policy; natural language processing; machine learning; classification; data use agreements

I. INTRODUCTION

Organizations derive policies from a wide variety of sources, such as business plans, laws, regulations, and contracts. As an extreme example, the United States Department of Defense has over 180 different policy documents and sources for its Trusted Global Information Grid [1]. While other organizations are not so overwhelmed with sources, they still must contend with manually processing documents to ensure their operations and systems are in compliance with all defined policies. To solve this issue, organizations need to implement a holistic approach to managing policy. Such an approach requires policy extraction from existing documents, an appropriate digital representation, and assurance the operating environment correctly implements policies.

The goal of our research is to assure compliance with established policies by ensuring policies in existing natural language texts are discovered, appropriately represented, and implemented.

To meet this goal, we propose a tool-based process, Natural Language Parsing for Policy (NLP4P), to allow organizations to utilize existing, uncontrolled natural language text to discover policies and represent those policies digitally. NLP4P will analyze documents to discover sentences and passages establishing a policy, extract out relevant policy elements, and then represent the policy in a digital format. For our initial work, we first need to determine which sentences establish policies and whether or not those policies can be enacted within a computer system. To evaluate this work we analyze four data use agreements (DUA).

A DUA is a legal contract among two or more parties that specifies what data is shared, who can access the data (authorizations), and for what purpose the data may be used (purposes). The contract may specify obligations, activities one party must perform (i.e., data must be encrypted), and constraints, activities one party must not perform (i.e., users shall not contact individuals identified in the data set). These authorizations, purposes, obligations, and constraints are a basis for policies for organizations. Additionally, DUAs may specify certain system requirements. Given the possibility to generate both policy and requirements from DUAs, we choose to classify sentences into one of five categories: policy, digital policy, functional requirement, non-functional requirement, and not applicable. For the purposes of this paper, we term policy to be statements that govern behavior within an organization and digital policy as policy that can be implemented within a computer system. We term functional requirements as specific functionality to be implemented by a system and non-functional requirements as characteristics of the system.

II. RELATED WORK

A. Controlled Natural Language

Other researchers have resolved converting natural language to and from machine policies by utilizing a controlled natural language (CNL). Schwitter [2] defines CNLs as "engineered subsets of natural languages whose grammar and vocabulary have been restricted in a systematic way in order to reduce both ambiguity and complexity of full natural languages." While CNLs provide consistent, semantic interpretations, CNLs limit authors and typically require language specific tools to stay within the constraints of the language. Project documents previously created cannot be used as inputs without processing the documents manually into the tools. Policy authored outside of tools must confirm to strict, limited grammars to be automatically parsed. Brodie et al. [3] used this approach in the SPARCLE Policy Workbench. Using their own natural language parser and a controlled grammar, they were effectively able to translate from natural language into formal policy. Users also responded favorably to their policy authoring tool. Recently, Shi and Chadwick [4] showed the improved usability of their CNL interface, but users were limited in the complexity of the policies that could be created as the interface did not support complex policies.

U.S. Army Research Office (ARO) under grant W911NF-08-1-0105

978-0-7695-4735-0/12 $26.00 © 2012 IEEE
DOI 10.1109/POLICY.2012.16

B. Data Use Agreements

Researchers have examined DUAs from several perspectives. Schmidt et al. [5] utilized a manual method to extract requirements from DUAs by analyzing the documents for commitments, privileges, and rights.[1] From their work, they identified contractual compliance requirements as well as identified issues associated with the current system in relation to the DUAs. Matteucci et al. [6] presented their approach for developing DUAs with a controlled natural language for agreements and a tool to support authoring of such agreements. They demonstrated the ability of the tool to author agreements and to detect potential policy conflicts.

III. Natural Language Parsing for Policy

We now present our proposed process, Natural Language Parsing for Policy (NLP4P), to classify sentences and then to extract elements from policy statements such that they can be digitally represented. To guide users through the process, we developed a tool to perform many of the tasks automatically and allow the user to make corrections when items were incorrectly classified.

A. Overview

For input, the process takes any natural language document that may serve as a source for policies and requirements. The process then follows four steps to classify sentences and represent applicable policies within a policy ontology.

1) Parse natural language into intermediate representation
2) Classify Sentences
3) Perform reference resolution
4) Convert intermediate representation to an ontology

Using an ontology as the knowledge representation provides a flexible format to manage policies in which we can detect conflicts and inconsistences.

B. Step 1: Parse Natural Language

The process begins by entering the text into the system, parsing the text and converting the parsed representation into NLP4P's intermediate representation (IR). The IR represents each sentence as directed graph where the vertices are words and the edges are the relationships between words.

The tool parses text with the Stanford Natural Language Parser and, for each sentence, outputs a graph in the Stanford Type Dependency Representation (STDR) [7]. We choose the STDR as it incorporates the sentence's structural information in a concise and usable format and can be readily converted to a Resource Description Framework (RDF) representation, which aids the conversion later to an ontology.

From the STDR generated by the parser, we create our IR as NLP4P needs to track additional attributes for the sentence and for each word. Fig. 1 shows the STDR for the sentence "A student may search and register for classes." Although, in general the IR can be considered a tree, situations exist (primarily due to conjunctions) in which a vertex has multiple parents. Vertexes correspond to words in the sentence and contain the word, the word's lemma and collapsed part of speech. Edges correspond to the relationship between two words (unchanged from Stanford's representation). Utilizing a pre-order traversal, the process creates the IR from the Stanford graph. As each vertex is created, we make two changes to the nodes. First, to avoid multiple versions of the same word, we use the lemma of the original word. Second, to avoid differences in the part of speech, we collapse the parts of speeches for all nouns, verbs, and adjectives to their base category. For example, we treat all plural nouns and proper nouns as just nouns. Similarly, verbs with different tenses are treated collectively as a single group.

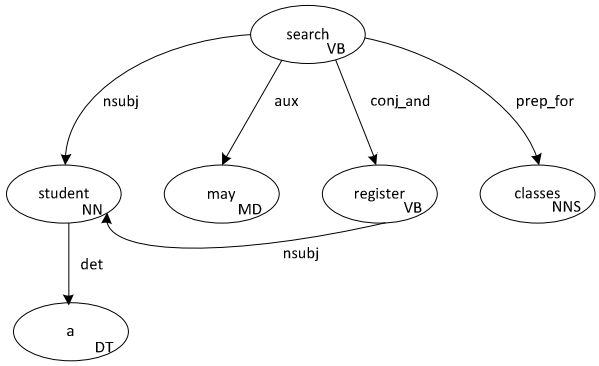

Figure 1. Stanford Collapsed Type Dependency Graph

C. Step 3: Classify Sentences

Once the tool completes the parsing and initial analysis of a sentence, a k-NN classification algorithm classifies each sentence into one of five categories: policy, digital policy, functional requirement, non-functional requirement, or not applicable. This classification is paramount to our process as the classification of "policy" or "digital policy" indicates the process will perform additional work to convert the sentence into a policy-based ontology. Sentences classified besides "not applicable" will appear on generated reports from the tool for use outside of the system.

A k-NN classifier predicts a classification by taking a majority vote of the existing classifications of the k nearest neighbors to the item under test. Thus, in our situation, to classify a sentence into one of the five categories, the classifier needs to find which sentences already classified are most similar to that sentence. k-NN classifiers use a distance metric to find the closest neighbors. This metric is the sum of the differences among the attributes used to determine the classification. Typically, Euclidean distance serves as a metric for numerical attributes while for nominal values, the distance is generally considered to be zero if both attribute values are the same or one if they differ. Our situation is more complex as we have a variable number of attributes to consider for each sentence based upon the sentence length. Additionally, certain words may be more closely related to one another than other words. As such, we need to utilize a custom distance metric to compute a value representing the difference between two sentences.

[1] They define "a commitment is an action that Party A pledges to Party B. A privilege is an action that Party A is entitled to perform that does not imply a commitment from any other party. A right is an action that Party A is entitled to perform that also implies a commitment from Party B to Party A."

Our distance metric is a modified version of the Levenshtein distance [9]. Rather than using the resulting number of edits to transform one string into another as the value as the Levenshtein distance does, our metric computes the number of word transformations to change one sentence into another. Rather than strictly using just zero or one as the difference between words, the metric uses the function defined in Fig. 2. The function first checks the structure of graph around each vertex to ensure it corresponds to other vertex. Next, the functions checks to see if the two vertices are the same (lemmas are equal). Then in line 7, the process checks to see if the two words are related through sets of cognitive synonyms (synsets) within WordNet [2] via semantic relationships (hypernym or hyponym). If a relationship value is found, then a value between 0.1 and 0.4 is returned based upon the number of relationships traversed. Finally, a default value of 0.6 is returned if none of the other conditions are met. In this situation, the vertices have an equivalent structure and part of speech and should be scored as closer together than two vertices differing in those attributes.

Once the classification is complete, the user may review the classification and provide correction as necessary through the tool.

```
computeVertexDistance(Vertex a, Vertex b)
 1: if a = NULL or b = NULL return 1
 2: if a.partOfSpeech <> b.partOfSpeech return 1
 3: if a.parentCount  <> b.parentCount return 1
 4: for each parent in a.parents
 5:   if not b.parents.contains(parent) return 1
 6: if a.lemma = b.lemma return 0
 7: wnValue = wordNetSynonyms(a.lemma,b.lemma)
 8: if wnValue > 0 return wnValue
12: return 0.6
```

Figure 2. Compute Vertex Distance Logic

D. Step 4: Perform Reference Resolution (future work)

Once a sentence has been classified as policy, we need to ensure the various elements required by the policy are present and, if not, to infer those properties from elsewhere in the text. For example, access control policies require at least a subject, an action, and a resource defined. It is feasible for one sentence to specify the action and resource, but the subject identified in another sentence. In work to be performed, the tool will be enhanced to classify the type of policy for the sentence, extract relevant attributes present, and then to resolve any missing policy elements.

E. Step 5: Convert Intermediate Representation to an Ontology (future work)

We will also modify the tool to convert the IR into an ontology. Once in this format, we can apply reasoning to check for conflicts and then convert to other formats such as eXtensible Access Control Markup Language (XACML) for deployment into the environment.

IV. Evaluation

To evaluate our process, we used four DUAs from a real system currently under development[3]. Text files were created from the Microsoft Word or Adobe PDF files and then imported directly into the tool without further changes. The first author executed the tool against the four files, checked the appropriate classification against his knowledge of the system, and applied corrections as necessary to ensure all sentences were correctly classified. The 336 sentences in the four files were classified as follows:

- 89 were policy statements, such as "Recipient shall not sell, rent or commercialize data in any manner to any person or entity.
- 3 were digital policy statements, such as "Access to the data will be restricted to persons expressly named and authorized by this Data Use Agreement."
- 29 were functional requirements, such as "Prospectively use this tool to identify unexpected increases in adverse outcomes in time and space."
- 8 were non-functional requirements, such as "Data will be encrypted at rest and in transit to the system."
- 207 were not applicable, such as "The data providers have made a substantial and long-term contribution in establishing and maintaining a database of high quality."

To compare the results, we used recall, precision, F_1 measure, and accuracy. To compute these values, we first need to categorize the classifier's predictions into three categories for each classification value. True positives (TP) are correct predictions. False positives (FP) are predictions in which the sentence of another classification is classified as the one under evaluation. False negatives (FN) are predictions in which a sentence of the same classification under evaluation is placed into another classification. From these values, we define precision (P) as the proportion of corrected 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_1 measure is the harmonic mean of precision and recall, giving an equal weight to both elements: $F_1 = 2 \times \frac{P \times R}{P + R}$. For precision, recall, and the F_1 measure we present a weighted average of these results. Accuracy is the proportion of correct classifications against all classifications. From a policy perspective, high values for both precision and recall are desired. Lower precision implies that the process will incorrectly mark sentences and require more human intervention to resolve. Lower recall implies that the classifier has missed policy and requirement statements from the source document.

Our first evaluation of the classifier used a stratified *n*-fold cross-validation in which data is randomly partitioned into *n* folds based upon each fold of approximately equal size and equal response classification. For each fold, the models are trained on the remaining folds and then the contents of the fold are used to test the model. 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 produces relatively low bias and variance. The cross-validation ensures that all sentences are used for training and each sentence is tested just once.

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

[3] The system owners preclude us from sharing more details.

To compare our classifier against other machine learning classifiers, we used several different options from WEKA [11] along with a weighted random model. To generate input data for the WEKA models, we flattened the IR graph into a comma separated list using a depth-first traversal and then executed a WEKA filter to convert all string values to nominal values. Our classifier significantly outperformed the other models as demonstrated in Table 1.

Additionally, we evaluated the classifier with three of the DUAs forming the training data set and the remaining DUA serving as test data set. The test was repeated such that each document served as the test data set once. The average accuracy result of 0.780 is just slightly less than the accuracy for the 10-fold validation. This decrease is explained by less sentences being available within the classifier and, hence, sentences are not matched to similar sentences as accurately.

TABLE I. TEST RESULTS OF STRATIFIED 10-FOLD CROSS VALIDATION

Model	Precision	Recall	F_1 Measure	Accuracy
Weighted Random	0.444	0.458	0.451	0.458
Naïve Bayes	0.717	0.720	0.681	0.720
BFTree	0.725	0.729	0.713	0.729
Ridor	0.602	0.643	0.619	0.643
IB1	0.709	0.705	0.707	0.705
NLP4P *k*-NN (*k*=1)	0.819	0.810	0.812	0.809

Next, we examined whether it was appropriate for the classifier to even return a value. Sentences do exist without any close neighbors in the classifier, and the resulting classification amounts to a weighted random guess. To determine whether or not classification results should be utilized, we computed a threshold value as a ratio of the calculated distance to the neighbors compared to the number of vertices in the sentence. So if the computed distance to its neighbors was 7.0 and the sentence had 10 vertices, we would only accept the classifier's answer if the threshold was set at 0.70 or higher. As can be seen in Table II, low threshold values (T) resulted in significantly higher values for the F_1 measure and accuracy with the disadvantage that a substantial number of sentences were not automatically classified.

TABLE II. THRESHOLD RESPONSE RESULTS

T	Stratified 10-Fold			Document Fold		
	% Answered	*F_1 Measure*	*Accuracy*	*% Answered*	*F_1 Measure*	*Accuracy*
0.6	43%	1.000	1.000	40%	1.000	1.000
0.7	49%	0.969	0.970	43%	1.000	1.000
0.8	78%	0.867	0.863	72%	0.866	0.863
0.9	99%	0.813	0.800	99%	0.782	0.778
1.0	100%	0.812	0.809	100%	0.784	0.780

V. CONCLUDING REMARKS

In evaluating the process and the tool, we only examined four DUAs for a specific system. Our results most likely are higher than if the DUAs came from different systems or projects. While the four DUAs are different, evidence exists where one of the DUAs has been used a starting point for another. Additionally, two of the DUAs could also be traced back to a template available on the internet.

We plan to complete the NLP4P process in that the process can generate digital policy representations in a policy-based ontology. The process will be enhanced to perform more analysis of the IR to detect conditions and other patterns. We will continue to explore other documents types as policy sources. Our process and its associated tool that will guide policy writers and requirements analysts in extracting policy and requirement statements from existing natural language documents. Preliminary results from examining DUAs show classification into policies and requirements is feasible.

REFERENCES

[1] "Identity & Information Assurance- Related Policies and Issuances," 2012. [Online]. http://iac.dtic.mil/iatac/download/ia_policychart.pdf. [Accessed: 01-Oct-2012].

[2] R. Schwitter, "Controlled Natural Languages for Knowledge Representation," in *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010, pp. 1113–1121.

[3] C. a. Brodie, C.-M. Karat, and J. Karat, "An Empirical Study of Natural Language Parsing of Privacy Policy Rules Using the SPARCLE Policy Workbench," *Proceedings of the second symposium on Usable privacy and security - SOUPS '06*, p. 8, 2006.

[4] L. Shi and D. Chadwick, "A Controlled Natural Language Interface for Authoring Access Control Policies," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, 2011, pp. 1524-1530.

[5] J. Y. Schmidt, A. I. Antón, L. Williams, and P. N. Otto, "The Role of Data Use Agreements in Specifying Legally Compliant Software Requirements," in *Fourth International Workshop on Requirements Engineering and Law*, 2011, no. Relaw, pp. 15-18.

[6] I. Matteucci, M. Petrocchi, M. L. Sbodio, and L. Wiegand, "A Design Phase for Data Sharing Agreements," in *6th DPM International Workshop on Data Privacy Management*, 2011.

[7] M.-C. de Marneffe, B. MacCartney, and C. Manning, "Generating Typed Dependency Parses from Phrase Structure Parses," *Proceedings of Language Resources and Evaluation*, pp. 449–454, 2006.

[8] R. M. W. Dixon, *A Semantic Approach to English Grammar*, Second. Oxford University Press, USA, 2005, p. 543.

[9] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[10] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2011, p. 744.

[11] M. Hall et al., "The WEKA Data Mining Software : An Update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10-18, 2009.