Information Extraction to Facilitate Translation of Natural Language Legislation

by

Samuel Wang

B.S. Computer Science
B.S. Mathematics
Carnegie Mellon University, 2008

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering at the Massachusetts Institute of Technology

February, 2011

© 2011 Massachusetts Institute of Technology All rights reserved.

Author: 

Department of Electrical Engineering and Computer Science
January 28, 2011

Certified by:  

Hal Abelson
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by:  

Professor Terry P. Orlando
Chairman, Department Committee on Graduate Students
Abstract

There is a large body of existing legislation and policies that govern how government organizations and corporations can share information. Since these rules are generally expressed in natural language, it is difficult and labor intensive to verify whether or not data sharing events are compliant with the relevant policies. This work aims to develop a natural language processing framework that automates significant portions of this translation process, so legal policies are more accessible to existing automated reasoning systems.

Even though these laws are expressed in natural language, for this very specific domain, only a handful of sentence structures are actually used to convey logic. This structure can be exploited so that the program can automatically detect who the actor, action, object, and conditions are for each rule. In addition, once the structure of a rule is identified, similar rules can be presented to the user. If integrated into an authoring environment, this will allow the user to reuse previously translated rules as templates to translate novel rules more easily, independent of the target language for translation.

A body of 315 real-world rules from 12 legal sources was collected and annotated for this project. Cross-validation experiments were conducted on this annotated data set, and the developed system was successful in identifying the underlying rule structure 43% of the time, and annotating the underlying tokens with recall of .66 and precision of .66. In addition, for 70% of the rules in each test set, the underlying rule structure had been seen in the training set. This suggests that the hypothesis that rules can only be expressed in a limited number of ways is probable.
Acknowledgments

I would like to thank the following people for their assistance in making this work possible:

Hal Abelson, my research advisor, for advising me over the last two years and providing constant guidance for the context of this work.

K. Krasnow Waterman, for giving me the original inspiration for this work and providing valuable advice on the legal side of the problem.

Joe Pato, for proofreading a version of this document and providing many valuable comments.

The other members of the Decentralized Information Group at MIT, who have given assistance and feedback for this work over the last 2 years.

Everyone else I have consulted at MIT and elsewhere for advice regarding the feasibility and practicality of this project.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Introduction</strong></td>
<td>11</td>
</tr>
<tr>
<td>1.1 Problem</td>
<td>11</td>
</tr>
<tr>
<td>1.2 Proposed Solution</td>
<td>12</td>
</tr>
<tr>
<td>1.3 Scenario</td>
<td>13</td>
</tr>
<tr>
<td><strong>2 Related Work</strong></td>
<td>17</td>
</tr>
<tr>
<td>2.1 Policy Management</td>
<td>17</td>
</tr>
<tr>
<td>2.2 Representing Law</td>
<td>18</td>
</tr>
<tr>
<td>2.3 Natural Language Processing</td>
<td>18</td>
</tr>
<tr>
<td><strong>3 Data Collection</strong></td>
<td>21</td>
</tr>
<tr>
<td>3.1 Data Crawling</td>
<td>21</td>
</tr>
<tr>
<td>3.2 Data Preprocessing</td>
<td>23</td>
</tr>
<tr>
<td>3.3 Data Annotation</td>
<td>27</td>
</tr>
<tr>
<td>3.4 Limitations of the Data Model</td>
<td>29</td>
</tr>
<tr>
<td>3.4.1 Dependencies Between Rules</td>
<td>29</td>
</tr>
<tr>
<td>3.4.2 Inability to Reduce to Individual Rules</td>
<td>30</td>
</tr>
<tr>
<td><strong>4 Implementation</strong></td>
<td>33</td>
</tr>
<tr>
<td>4.1 Implementation Overview</td>
<td>33</td>
</tr>
<tr>
<td>4.2 Model Training</td>
<td>36</td>
</tr>
<tr>
<td>4.2.1 Template Construction</td>
<td>36</td>
</tr>
<tr>
<td>4.2.2 Language Models</td>
<td>37</td>
</tr>
<tr>
<td>4.2.3 Training Outcome</td>
<td>40</td>
</tr>
<tr>
<td>4.3 Annotating Novel Rules</td>
<td>41</td>
</tr>
<tr>
<td>4.3.1 Annotating Conditional Clauses</td>
<td>42</td>
</tr>
<tr>
<td>4.3.2 Template Matching</td>
<td>45</td>
</tr>
<tr>
<td>4.3.3 Extraction Token Classification</td>
<td>46</td>
</tr>
<tr>
<td>4.3.4 Match Scoring</td>
<td>49</td>
</tr>
<tr>
<td>4.3.5 Summary</td>
<td>50</td>
</tr>
<tr>
<td><strong>5 Evaluation</strong></td>
<td>53</td>
</tr>
<tr>
<td>5.1 Experimental Setup</td>
<td>53</td>
</tr>
<tr>
<td>5.2 Evaluation of Conditional Extraction</td>
<td>54</td>
</tr>
<tr>
<td>5.2.1 Analysis of Head Conditional Extraction</td>
<td>55</td>
</tr>
</tbody>
</table>
List of Figures

3.1 Raw Legislation from Idaho 67-3008 .................................................. 24
3.2 Raw Legislation from Arizona 41-1750 ............................................. 25
3.3 Raw Legislation from Massachusetts Part I Title II Chapter 6 Section 172 26
3.4 Figure 3.2 with preprocessing applied ............................................. 27
3.5 A sample annotated rule from Maryland 12.15.01.11 .......................... 28
3.6 An example rule pair that does not fall into our data model from Maryland 29
3.7 A sample from Wyoming Title 7 Chapter 19 where hierarchy is key to the semantic meaning of the rule .............................................. 30

4.1 Architecture Overview ................................................................. 34
4.2 Two annotated rules that share the same template ............................. 34
4.3 Simplified example of the system approach ...................................... 35
4.4 Sample template and the regular expression derived from it ................ 36
4.5 A novel rule with two possible template matches ............................... 37
4.6 Two annotated training samples, along with the part of speech for each token. JJ is an adjective, NN is a noun, NNS is a plural noun, VB is a verb, MD is a modal auxiliary, DT is a determiner ......................................................... 38
4.7 Computing the smoothed probability for a part of speech. In this formula, \textit{count(POS)} is the number of occurrences of the given part of speech, \textit{N} is the number of different parts of speech, and \textit{M} is the total number of tokens encountered in the training set ........................................................................................................ 39
4.8 Example where the model for ACTION in two different templates are different ................................................................. 40
4.9 Unannotated rule for the following section ....................................... 41
4.10 Pseudocode for evaluating an unknown input rule ............................. 42
4.11 Unannotated rule with heuristics extracting conditionals .................... 45
4.12 Matching the input rule against seen templates .................................. 46
4.13 Calculating the probability for each split point .................................. 47
4.14 Extracting classes after finding optimal partitions .............................. 48
4.15 Calculating the scores for matching each template against the novel rule ............................................................... 50
4.16 The final output of the system given the sample input ........................ 51
4.17 The final output of the system given the rule from IIR 28 CFR Part 23 .... 52

5.1 Rule where head conditional extraction fails, from US Code Title 42 Chapter 6A Subchapter III-A Part D 290dd2 .................................................. 55
5.2 Rule where tail conditional extraction fails, from Arizona 41-1750. ........ 56
5.3 Two template matches for a rule from Washington RCW 10.97.050 ......... 59
5.4 Two template matches for a rule from Kentucky 502 KAR 30:060 . . . . . . . . . . 60
5.5 The one rule instance that derived the correctly matching template . . . . . . . . . . 60
## List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Table Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Table of law sources used in this work</td>
<td>22</td>
</tr>
<tr>
<td>3.2</td>
<td>A list of the possible annotations for the components of each rule</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Frequency of each annotation in the training set</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>A sample of templates along with their frequencies</td>
<td>36</td>
</tr>
<tr>
<td>4.2</td>
<td>The probabilities assigned to each part of speech for each annotation class</td>
<td>38</td>
</tr>
<tr>
<td>4.3</td>
<td>The probabilities assigned to each part of speech for each annotation class, after smoothing is applied with $\gamma = 0.05$</td>
<td>39</td>
</tr>
<tr>
<td>4.4</td>
<td>A list of templates with conditionals and their frequencies, as seen in the corpus annotated in Chapter 3. Note that the occurrence of each of these is very small.</td>
<td>43</td>
</tr>
<tr>
<td>4.5</td>
<td>List of phrases that signal an upcoming conditional phrase</td>
<td>44</td>
</tr>
<tr>
<td>4.6</td>
<td>The probability function for (annotation class, POS) for the template {ACTOR} may {ACTION} {OBJECT} to {RECIPIENT}</td>
<td>47</td>
</tr>
<tr>
<td>5.1</td>
<td>Evaluation scores for the heuristic-based conditional tagging mechanism</td>
<td>54</td>
</tr>
<tr>
<td>5.2</td>
<td>Results for template matching</td>
<td>57</td>
</tr>
<tr>
<td>5.3</td>
<td>Rule templates that are unique in the training set</td>
<td>57</td>
</tr>
<tr>
<td>5.4</td>
<td>Results for template matching when conditional extraction heuristics are disabled</td>
<td>61</td>
</tr>
<tr>
<td>5.5</td>
<td>Results for individual token annotations</td>
<td>62</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Problem

Historically, the problem of information handling policy compliance within organizations is not solved using technology. Rather, policies are stated by lawyers or executives, and it is assumed that employees will abide faithfully to the relevant rules while doing their work. However, an organization of this nature has limited ability to audit old transactions if rules are broken, as it will require human intervention to piece together which rules were broken by which users at what times.

Specifically, past work we have done [14] has shown that this is a problem in the sphere of government information sharing. Since 9/11, it has been acknowledged that there was an insufficient amount of information sharing between different US intelligence organizations, and the federal government has established programs to facilitate the increased sharing of intelligence information [2]. However, there are a myriad of state and federal laws that apply to sharing of this nature. The laws vary based on the specific information being transmitted; for example, there are laws that apply specifically to juvenile records, or laws that apply only to mental health records. In addition, each state and organization may have policies that apply to information sent to or from that state or organization. The analysts whose job it is to facilitate these information sharing transactions
should not have to memorize the laws for each state that they send or receive intelligence from. In this case, we really want to serialize the applicable policies into a computer-understandable format, so that software will be able to assist in judging compliance. In our prior work [14], we have prototyped the various pieces of such a system, and demonstrated that it could be integrated into a data-sharing environment successfully.

However, one challenge of our prior work was that it required a team of both a lawyer and a computer programmer to accurately encode real-life legislation into a formal policy language that computers could understand. This is a very labor-intensive process, and required the coordination of lawyers as well as computer programmers to accomplish.

1.2 Proposed Solution

Natural language processing techniques have not advanced to the point where it is feasible for a computer to do this complicated translation completely unassisted. The aim of this work is to provide a system that will facilitate this translation process. The key observation is that in reality, there are only a few natural language blueprints that are used to express rules in this domain. In general, the actors, objects, actions, and conditionals may change from rule to rule, but there are a very limited number of ways that these rules are fundamentally expressed via language. The idea is to divine these templates that rules are based on, and then extract the salient parts of each rule. In this way, the system can process a series of natural language rules, and for each rule, present the user with the general template that the rule is based on, as well as extracting the list of actors, actions, objects, and conditionals that are present in the given rule. The idea is that if users have translated other rules with the same rule template, they can simply reuse the formal language code that they have already written and simply replace the specific actors and actions. This facilitates translation in a faster way than was possible before.

There was a conscious decision to not tie the system to any particular formal language. In reality, different organizations are all going to use different systems to facilitate this type of access
control or accountability. A system built with one target language in mind has limited applicability. The hypothesis is that, regardless of the formal language, there will be a modular way to write rules in the formal language, where snippets of code can be reused in different contexts with different variables. So, if the association is made that some template of natural language rules is associated with some snippet of formal language code, the authoring platform that the user is using can integrate this knowledge, presenting the user with the appropriate formal language snippet for each rule to be translated. In this way, the system is adaptable for any formal language system.

This work began with collecting 315 rules from 12 different federal and state sources. Rules were annotated by hand to produce a training set of data. Afterwards, a system was written that derives the template from each rule in the training set, aggregates those templates together, and is then able to match those templates against novel untagged rules fed as inputs to the system. Standard cross-validation evaluation was run on the annotated rules, and produced results that suggested that the system is robust enough to assist users with this translation process. In particular, for each cross-validation test set, 70% of the rule templates in the test set had been seen before in the training set. This indicates that there are only a limited number of ways rules are expressed in this domain.

1.3 Scenario

Suppose a lawyer, Alice, wants to use the system to help translating some natural language rules. Alice would be a lawyer or policy-maker in a government organization or private industry, and wishes to ensure that users at her organization are compliant with the legislation or regulation that Alice is translating. The proprietary policy-enforcement system at Alice’s organization requires rules written in some formal policy language in order to be used, and Alice wants help translating the substantial amount of natural-language policies into the formal language.

The following is a rule that Alice wants to translate:
As long as the agent lawfully collected the information, the criminal justice agency, if given a request in writing beforehand, may disseminate information about an individual to another agency.

When the unannotated rule is presented to the system, the system will produce a sorted list of candidates, where each candidate consists of two separate components:

1. A list of extractions, with each extraction consisting of a type and a sequence of tokens from the rule.

2. A list of similar natural language rules that the system has seen before.

For the sample input, a candidate output is given below:

ACTOR = the criminal justice agency
ACTION = disseminate
OBJECT = information about an individual
RECIPIENT = another agency
CONDITIONAL = the agent lawfully collected the information
CONDITIONAL = given a request in writing beforehand

Similar rules:
As long as the analyst is certified by the board, the analyst, if requested by another employee, may give criminal history information to the requesting employee.
...

The similar rule presented by the system would have been a rule previously encountered by Alice, and would have been both annotated and translated by her prior to being given to the system.

To complete the translation of the novel rule, Alice would take extracted tokens of the novel rule presented by the system and convert them into whatever vocabulary is used by the formal
language in question. She would then substitute those into the formal rule snippet provided by the similar rule.

pattern {
  actor: criminal_justice_agency
  action: disseminate
  object: information_about_individual
  recipient: another_agency
} if {
  if: agent_lawfully_collected_the_information AND
  if: given_a_request_in_writing_beforehand
} then {
  compliant
} else {
  non-compliant
}

In order to train the system to give the behavior described above, Alice would have had to annotate and translate a number of natural language rules.

As long as CONDITIONAL{the analyst is certified by the board}, ACTOR{the analyst}, if CONDITIONAL{requested by another employee}, may ACTION{give} OBJECT{criminal history information} to RECIPIENT{the requesting employee}.

pattern {
  actor: analyst
  action: give
  object: criminal_history_information
  recipient: requesting_employee
} if {
  if: analyst_certified
  if: requested_by_employee
} then {
  compliant
} else {
  non-compliant
}

In this manner, Alice will only need to do the translation for each broad type of natural language
rule once into the formal language; each similar natural language rules will prompt Alice with the previously translated rule, and it will be easy for Alice to generalize the original translation.

The remainder of this thesis will be organized in the following manner: Chapter 2 will focus on related work for this project, Chapter 3 will explain the challenges for collecting data to train and evaluate such a system, Chapter 4 will describe the implementation of the system, Chapter 5 will document how the system was evaluated and describe potential improvements that could be made, and finally Chapter 6 will conclude with a description of future work and a summary.
Chapter 2

Related Work

This work is positioned at the intersection of previous work in policy management systems, representing law in formal languages, and natural language processing.

2.1 Policy Management

There are existing systems that attempt to solve the similar problem of rule authoring [7]. SPARCLE’s target end-user is a non-technical user in an enterprise who has a mental picture of the policies they want to express. The system allows the user to input guided natural language; the system presents a template rule that the user can fill in, which the system can then transform into machine language. Oracle has also released a similar system [6], which is also geared towards rule authoring rather than translation.

However, the problem they consider is significantly different, in that they take the stance that users have the flexibility of rephrasing rules in a way that the machine can accurately understand them. In our case, the rules have been decided upon in a manner that the user has no influence on, and the algorithm has to adapt to the language already in use. It is unlikely that feeding an existing natural language rule into one of these systems will produce acceptable output without rewriting the input in some way, which is not effective for translating a large number of rules.
2.2 Representing Law

There has been some existing work in marking up legal text with metadata [1] [5] [4]. These efforts generally use domain-specific rules that apply to the text to get their results, and focus on law in general rather than the small domain of information sharing policies dealt with in this work. Due to their broad focus, it appears that the existing work in this domain is not yet at the level where the analyzed text is suitable to be fed into an automated reasoner. It could be that the annotations applied by these existing systems could be applied to this domain, and the steps taken in this system to group rules together and provide similar rules could be a refinement on the existing systems.

2.3 Natural Language Processing

One of the primary inspirations for this work came from the work by Wang et al [13], which attacks the completely different problem of set expansion using the internet. The idea in their work is that by looking at the left and right context of things known to be in a set, one can extract elements of the same set by looking for other words surrounded by the same context. The idea is similar to the idea explored in this work; the idea is that law is so regimented that simply looking at the raw context surrounding the actor, action, or object in a rule is enough to extract from other rules.

There has been significant work done in the problem of semantic parsing, which seeks to automatically translate natural language strings into a formal language counterpart [12] [8] [15]. Each of these systems tries to leverage different natural language processing technologies to automatically translate a natural language string (“what states border texas”) into some formal language ($\lambda x.\text{state}(x) \land \text{borders}(x, \text{texas})$). One of the primary reasons why these systems are not suitable for this task is that in the domain for each of them, there is a fairly limited vocabulary to express objects and relations. For example, in the domain for the example given, there are a finite number of states and a one can only conceive of a handful of relations between them. However, for this domain, the set of objects that the law may want to talk about is vast, there may be variation in the way that the same object is talked about in the law, and there may be limitations imposed by the
formal language on the vocabulary for translation. In addition, the formal rules language in this case may not be expressible by a context-free grammar, which is required by many of the systems in question. For these reasons, it seems like the problem being expressed here is too challenging for the state-of-the-art semantic parsing systems.

There are also existing systems in named-entity recognition, which attempts to annotate natural language strings with a finite set of tags, after being trained on a set of pre-annotated data [11]. The primary problem with these systems is that even though they could potentially annotate a string with the appropriate tags, as we want, they do not do the second half of prompting the user with similar rules that have potentially already been translated. Again, similar to the work done annotating existing legal sources, it would be useful in future work to see if the techniques for aggregating rules together and aiding users in translation could be a refinement of existing named-entity recognition systems.
Chapter 3

Data Collection

The fundamental research portion of this thesis is questioning whether or not it is feasible to facilitate translation of natural-language rules into a formal language by prompting a human translator. However, to actually investigate this problem, data has to be collected and annotated from various legal sources. Completing this part of the problem is a software engineering task outside the scope of this thesis; to automate it, a specialized crawler would have to be written that handles many different input sources. Sections 3.1 and 3.2 explain how this process is done by hand, and details all of the components that would be necessary in order to automate it.

After preprocessing the input, the input rules are annotated by hand so that the automated annotator can learn how to parse rules automatically, and so the annotations produced by the system can be evaluated. The annotation process is detailed in Section 3.3.

Finally, Section 3.4 details the limitations of the model given by this chapter, and what steps could be taken for a more comprehensive data model.

3.1 Data Crawling

Legislation was collected from the websites of various federal and state organizations. Specifically, this work focuses on pieces of legislation that detail the rules that organizations must follow in order to share particular pieces of information, as that was the focus of the prior work we had done.
Twelve distinct pieces of legislation were used, which culminated in 315 rules being extracted and annotated from those pieces of legislation (see Table 3.1).

<table>
<thead>
<tr>
<th>Legislation Source</th>
<th>Title</th>
<th>Number of Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama Section 41-10-399</td>
<td>Confidentiality of data; inapplicability of public record laws; use of data by authority; requirements for contractual agreements for supercomputer services.</td>
<td>9</td>
</tr>
<tr>
<td>Arizona Section 41-1750</td>
<td>Central state repository; department of public safety; duties; funds; accounts; definitions</td>
<td>75</td>
</tr>
<tr>
<td>Idaho Section 67-3008</td>
<td>Release of criminal history record information</td>
<td>16</td>
</tr>
<tr>
<td>Institute for Intergovernmental Research 28 CFR Part 23</td>
<td>Criminal intelligence systems operating policies</td>
<td>35</td>
</tr>
<tr>
<td>Kentucky 502 KAR 30:060</td>
<td>Dissemination of criminal history record information</td>
<td>23</td>
</tr>
<tr>
<td>Maryland 12.15.01.11</td>
<td>Dissemination of CHRI - Criminal Justice Agency</td>
<td>10</td>
</tr>
<tr>
<td>Massachusetts 6-172</td>
<td>Dissemination of record information; certification; eligibility for access; scope of inquiry; listing; access limited; rules; use of information</td>
<td>24</td>
</tr>
<tr>
<td>Montana 44-5-303</td>
<td>Dissemination of confidential criminal justice information – procedure for dissemination through court</td>
<td>18</td>
</tr>
<tr>
<td>U.S. Code Title 18 Part IV Chapter 403</td>
<td>Use of juvenile records</td>
<td>10</td>
</tr>
<tr>
<td>U.S. Code Title 42 Chapter 6A Subchapter III-A Part D 290dd2</td>
<td>Confidentiality of records (Mental Health)</td>
<td>11</td>
</tr>
<tr>
<td>U.S. Code Title 5 Part I Chapter 5 Subchapter II 552a</td>
<td>Records maintained on individuals</td>
<td>22</td>
</tr>
<tr>
<td>Washington State 10.97.050</td>
<td>Restricted, unrestricted information Records</td>
<td>15</td>
</tr>
<tr>
<td>Wyoming Title 7 Chapter 19</td>
<td>Criminal History Records</td>
<td>47</td>
</tr>
</tbody>
</table>

Table 3.1: Table of law sources used in this work

Collecting this data demonstrated many of the difficulties that would be faced for a large-scale automated translation of legislation in this domain:

1. There is no centralized repository that collects all of the legislation from each separate or-
ganization. Each state generally maintains their own legislation on their own website with their own unique web interface. Thus, this stymies any effort to systematically gather all of the legislation of this type with minimal human intervention without trying to crawl each site via brute force.

2. There is no universal format that each source publishes their data in. Some organizations publish their legislation in plaintext, others in an HTML document, and others as PDFs. In addition, for sites that publish in HTML, there are often additional UI boilerplate elements on the page that could confuse an automated crawler.

3. For some pieces of legislation, states may conflate other policies with the rules that we are interested in. For example, in Arizona 41-1750, the legislation lists the rules that govern information sharing, but also lists a series of definitions used in the document. Thus, any solution that fully automates this process needs to find the relevant rules in a document that potentially has information that is unnecessary for this specific translation problem.

Each of the problems above can be solved technically with robust enough crawling software. However, for the purposes of this work, the contents of each piece of legislation was manually retrieved from the relevant pages, and unnecessary text was manually cropped out. The raw input for the next section is a series of plaintext files for each source that contains only the pertinent rules.

3.2 Data Preprocessing

From the sample of legal sources that were selected for training, it is clear that there is no uniform method for writing policies that all states and organizations share. The following examples will demonstrate that it is unsuitable to use the raw legislation as a direct input into the system, and that some amount of preprocessing is necessary so that data from various sources can be used in the same system. In particular, real world legislation uses lists and nested hierarchies to express a
hierarchy to the reader, but often times this structure is unnecessary for translation, or is not present in some data sources.

Example: Idaho

(3) Judicial review of the departments denial of a request for records shall be in accordance with the provisions of section 9-343, Idaho Code.

(4) A request for a criminal history record by a criminal justice agency or a court shall take precedence over all other requests. The department shall adopt rules to set forth the manner by which criminal justice agencies and courts without direct access to the public safety and security information system established by section 19-5202, Idaho Code, may request Idaho criminal history record information.

Figure 3.1: Raw Legislation from Idaho 67-3008

As exemplified in this snippet of legislation from Idaho (see Figure 3.1), sometimes the list format does not actually convey any semantic meaning. That is, the list is strictly used as a way to enumerate the rules in the law, but does not imply any relation between the various rules to each other. In addition, the rule set forth in (4) actually has two completely separate components; the first sentence makes a statement about the ordering of requests, while the second is a directive to the department for rules that should be adopted. Both of these rules should be handled separately by the translation, as there is no logical reason to consider both simultaneously while translating. It is clear that in this case that the leading numbers should be discarded as input from the system as part of the preprocessing, as they give no information about an appropriate translation.

Example: Arizona

The outline form used in this law from Arizona (see Figure 3.2) conveys that "the department" has a series of responsibilities, and this hierarchical structure is easy for humans to understand. However, it would take significant effort to special case this type of input for an automated crawler to understand; it would need to understand that the leading numbers should be dropped, and that the sentence "The department shall" should be prepended to the head of each sentence in the nested list.
The department is responsible for the effective operation of the central state repository in order to collect, store and disseminate complete and accurate Arizona criminal history records and related criminal justice information. The department shall:

1. Procure from all criminal justice agencies in this state accurate and complete personal identification data, fingerprints, charges, process control numbers and dispositions and such other information as may be pertinent to all persons who have been charged with, arrested for, convicted of or summoned to court as a criminal defendant for a felony offense or an offense involving domestic violence as defined in section 13-3601 or a violation of title 13, chapter 14 or title 28, chapter 4.

2. Collect information concerning the number and nature of offenses known to have been committed in this state and of the legal steps taken in connection with these offenses, such other information that is useful in the study of crime and in the administration of criminal justice and all other information deemed necessary to operate the statewide uniform crime reporting program and to cooperate with the federal government uniform crime reporting program.

Figure 3.2: Raw Legislation from Arizona 41-1750

Applying this operation would produce a series of self-contained rules like “The department shall procure from...” and “The department shall collection information...”. One could write a parser that directly applied these transformations to the given input, but conflated with the problems cited in Section 3.1, it would require a more sophisticated preprocessor to do this type of expansion automatically.

**Example: Massachusetts**

In this law from Massachusetts (see Figure 3.3), there is no hierarchical or list structure to delimit each individual rule. The entire legislation is expressed as a series of paragraphs, and each sentence can be considered as a self-standing rule. This demonstrates that a system that tries to annotate legislation like this cannot rely on the existence of lists or hierarchies in order to translate.

**Data Serialization**

The preceding examples seeks to illustrate the fact that if the system wants to be able to accept inputs from all of these sources, the system needs to either treat each case separately and have
No agency or individual shall have access to criminal offender record information under clause (c), unless the board, by a two-thirds majority of the members present and voting, determines and certifies that the public interest in disseminating such information to such party clearly outweighs the interest in security and privacy. The extent of access to such information under clause (c) shall also be determined by such a two-thirds majority vote of the board. Certification for access under clause (c) may be either access to information relating to a specific identifiable individual, or individuals, on a single occasion; or a general grant of access for a specified period of time not to exceed two years. A general grant of access need not relate to a request for access by the party or parties to be certified. Except as otherwise provided in this paragraph the procedure and requirements for certifying agencies and individuals under clause (c) shall be according to the provisions of the preceding paragraphs of this section.

Figure 3.3: Raw Legislation from Massachusetts Part I Title II Chapter 6 Section 172
The department is responsible for the effective operation of the central state repository in order to collect, store and disseminate complete and accurate Arizona criminal history records and related criminal justice information.

The department shall procure from all criminal justice agencies in this state accurate and complete personal identification data, fingerprints, charges, process control numbers and dispositions and such other information as may be pertinent to all persons who have been charged with, arrested for, convicted of or summoned to court as a criminal defendant for a felony offense or an offense involving domestic violence as defined in section 13-3601 or a violation of title 13, chapter 14 or title 28, chapter 4.

The department shall collect information concerning the number and nature of offenses known to have been committed in this state and of the legal steps taken in connection with these offenses, such other information that is useful in the study of crime and in the administration of criminal justice and all other information deemed necessary to operate the statewide uniform crime reporting program and to cooperate with the federal government uniform crime reporting program.

Figure 3.4: Figure 3.2 with preprocessing applied

3.3 Data Annotation

After converting each piece of legislation into a list of independent rules, annotations were applied by hand to each rule. Almost all rules dictates an obligation that an actor has, or conditions that need to apply in order for an action to happen. Because of the limited scope of rules in this domain, each rule can be characterized by annotations into only a handful of classes (see Table 3.2).

First, each rule is split into tokens on whitespace and apostrophes. For each rule, if any of the annotation classes are present in the rule, tokens of the rule are annotated with that particular class (see Figure 3.5). Any given token can not be annotated with two different classes (annotations are mutually exclusive), and some tokens are not annotated at all. A rule can be annotated by the same class twice (for example, if there are two conditions that need to be met before an action can take place, or if two objects are transmitted).

In Table 3.3, the frequency of each annotation is listed. The most used classes were Actions and Objects; every rule revolves around objects in question and what can and cannot be done to them. Slightly less used were actors; the action used in every rule is usually associated with an actor performing the action (e.g. “A person may disseminate...”), but often objects and actions can be re-
Annotation Class | Definition | Examples
---|---|---
Actor | The person or organization conducting the action | A criminal justice agency, the disseminating agent
Action | The action that the actor does, or the action that is applied to the object | disseminate, request, verify
Object | The object or entity being manipulated by the actor | Criminal history record information, a record or log, the information
Recipient | The receiver of the object in question | another criminal justice agency, a member of the public
Conditional | Requirements or conditions that need to be met before the action can take place | it has a need for the information in the performance of its function, any purpose associated with the administration of criminal justice

Table 3.2: A list of the possible annotations for the components of each rule

Subject to CONDITIONAL{the provisions of Regulation 12B,} ACTOR{the Central Repository and other criminal justice agencies} shall ACTION{disseminate} OBJECT{CHRI, be it conviction or nonconviction criminal history record information}, to RECIPIENT{a criminal justice agency} upon CONDITIONAL{a request made in accordance with applicable regulations adopted by the Secretary}.

Figure 3.5: A sample annotated rule from Maryland 12.15.01.11

ferred to with an implicit actor (e.g. “Criminal history record information may be disseminated...”), which may have been stated in an earlier rule or is implied by the document. Conditionals were present in a majority of rules; the remainder of rules generally implicitly condition the action based on the actor and the recipient (e.g. “An agent may share information to another agent...”). Finally, recipients were present in a fraction of the rules, leaving open the challenge of whether the rule truly means “anyone” or if it is somehow inferable from earlier rules or the document.
<table>
<thead>
<tr>
<th>Annotation Class</th>
<th>Number of Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>256</td>
</tr>
<tr>
<td>Action</td>
<td>324</td>
</tr>
<tr>
<td>Object</td>
<td>329</td>
</tr>
<tr>
<td>Recipient</td>
<td>79</td>
</tr>
<tr>
<td>Conditional</td>
<td>202</td>
</tr>
<tr>
<td>Number of rules</td>
<td>315</td>
</tr>
</tbody>
</table>

Table 3.3: Frequency of each annotation in the training set

3.4 Limitations of the Data Model

The process detailed in the first sections of this chapter are adequate for the majority of rules that were encountered while cultivating a training set for this project. However, there were exceptions where the model used fails; it is worthwhile to acknowledge these problems, and if sufficiently important, reevaluate the assumptions made in order to account for them in a future system.

3.4.1 Dependencies Between Rules

In arbitrary natural language, it is certainly the case that one thought or idea can be explained across multiple sentences or multiple paragraphs. In this specific domain, for the purpose of automating an access control system, it seems to be the case that the majority of rules can act atomically without referring to the sentences surrounding it. However, this model inadequately describes some specific cases.

Figure 3.6 shows, it shows a pair of rules that directly refer to each other. The second one refers to "disseminations under these conditions", clearly referring to the rule directly preceding it. However, this contextual information will be lost, as each rule is treated independently in our model. Even if our system correctly identifies the object in the second rule as "a log of each
This section does not apply if the receiving criminal justice agency demonstrates to a responsible official of the disseminating criminal justice agency or the Central Repository that a delay in the receipt of information from the Central Repository will unduly impede necessary action by the requesting criminal justice agency or will violate or materially impair a substantive right of the person about whom the information is needed.

However, the disseminating agency shall maintain a log of each dissemination under these conditions, showing the: (a) Date of dissemination; (b) Information disseminated; (c) Criminal justice agency to whom it was disseminated; and (d) Date of the dissemination.

Figure 3.6: An example rule pair that does not fall into our data model from Maryland 12.15.01.11 dissemination under these conditions”, it may be unclear to the human translator as to what “these conditions” entail. If the user is translating these rules in the same order as the original document, it will be evident to the user as to what “these conditions” the second rule is referring to, as it will be referring to the rule that they had just translated. However, this dependency is not encapsulated in the system itself, which could be problematic if very complicated rules require many sentences to translate.

3.4.2 Inability to Reduce to Individual Rules

(a) The following persons shall be required to submit to fingerprinting in order to obtain state and national criminal history record information:

(i) Employees of substitute care providers certified by the department of family services pursuant to W.S. 14-4-101 through 14-4-116;

...  

(xix) All persons applying for licensure to the Wyoming board of midwifery whose application or other information received by the board indicates that the applicant has or may have been convicted of a crime, and any licensee of the board of midwifery upon written request from the board of midwifery as part of an ongoing investigation of or disciplinary action against the licensee.

Figure 3.7: A sample from Wyoming Title 7 Chapter 19 where hierarchy is key to the semantic meaning of the rule

In the section on preprocessing (Section 3.2), it was detailed how hierarchies are removed in
order to make the input to the system uniform. However, in some cases it is difficult to remove the hierarchy present in the policy. In the rule detailed in Figure 3.7, there are 19 different groups of people that are required to submit to fingerprinting. It is not obvious how to create individual rules that capture the meaning present without fundamentally rewriting the rule. The rule would not make sense logically if we merely concatenated each object in the list to the original sentence; “the following persons” clearly refers to the list that follows. Each individual item in the list cannot stand alone as a rule without somehow adapting the fact that the individuals being described need to submit to fingerprinting. So, in this case, the model is inadequate to accurately describe the input; the user may get a translation for rule (a), but the user needs to discover without aid from the system exactly which persons are referred to by rule (a).
Chapter 4

Implementation

From the previous chapter, we have an annotated set of real-world rules, ready to be used to train and evaluate on. This chapter discusses the system that takes the annotated data as a training set, and produces a trained annotator. This chapter also discusses the algorithms that the trained annotator uses to process incoming novel rules, and the output produced by the trained annotator for the enduser.

4.1 Implementation Overview

The underlying assumption that this implementation takes is that there are only a handful of ways that rules are expressed in natural language in this domain, and it is possible to divine the structure of the rule by matching it against the structures that have been seen in the training set. After an appropriate match has been found, the system can extract the appropriate annotations, because the template will indicate which entities belong at which locations in the string. This is done by aggregating all of the underlying rule structures in the training data to produce a set of all rule structures that the annotator believes can be used to express natural language rules. Afterwards, novel rules are matched against each of these rule structures, and each match is scored based on how good of a fit it is for the input rule.

We define a rule template as an object that describes the underlying structure of a rule, without
taking into account the specifics of any particular rule. For example, in Figure 4.2, the two annotated rules both convey the same idea; namely that if a condition is met, the actor may do some action to a recipient. The specific training rules have completely different conditionals, actors, actions, objects, and recipients, but do not differ in the underlying natural language structure in terms of the ordering of the annotations and the words that are not part of any annotation ("If", "may", "to").

If CONDITIONAL{the receiving agency is certified}, ACTOR{the criminal justice agency} may ACTION{disseminate} OBJECT{criminal history record information} to RECIPIENT{another agency}.

If CONDITIONAL{the subject consents}, ACTOR{the central repository} may ACTION{share} OBJECT{personally identifiable information about that person} to RECIPIENT{the agency}.

Template: If {CONDITIONAL}, {ACTOR} may {ACTION} {OBJECT} to {RECIPIENT}
In the training phase, the system derives the underlying template for each rule and aggregates all of the templates to create a rule template set (Section 4.2.1). In addition, the system constructs a language model for the tokens that fall into each annotation class for each template (Section 4.2.2); with this information, the system can give a score for how well a template matches a novel rule. The trained system has a set of templates that are used to match against rules, as well as a language model for the tokens that belong in each annotation class for each template.

After training, when evaluating a novel rule, the trained annotator first uses a series of heuristics to try to identify common conditional phrases in the rule (Section 4.3.1). Afterwards, the annotator attempts to match each template encountered in training against the novel rule (Section 4.3.2). If a match is found, it scores the match based on how well the extracted terms match the model created during training (Section 4.3.4). In this way, for each novel input rule, the trained annotator produces a series of template matches along with a confidence score for each possible match (Section 4.3.5).
4.2 Model Training

4.2.1 Template Construction

The input to the training system is a corpus of annotated rules, as described in Section 3.3. Each annotated rule is converted into a template by removing all of the annotated portions of the rule and replacing them with a placeholder token that indicates what type of annotation belongs in the blank, as shown in Figure 4.2. This generates a string that consists of annotation class placeholders alongside regular natural-language tokens (see Table 4.1 for some sample templates). Each template is associated with the frequency of how often it occurs in the training data, as well as the annotated training rules that generated that template.

<table>
<thead>
<tr>
<th>Template</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ACTOR} {ACTION} {OBJECT}</td>
<td>10</td>
</tr>
<tr>
<td>{ACTOR} must {ACTION} {OBJECT}</td>
<td>6</td>
</tr>
<tr>
<td>{ACTOR} shall {ACTION} {OBJECT} to {RECIPIENT}</td>
<td>5</td>
</tr>
<tr>
<td>{ACTOR} may {ACTION} {OBJECT}</td>
<td>5</td>
</tr>
<tr>
<td>{OBJECT} shall {ACTION} on {CONDITIONAL}</td>
<td>5</td>
</tr>
<tr>
<td>{ACTOR} shall {ACTION} to {RECIPIENT} {OBJECT}</td>
<td>4</td>
</tr>
<tr>
<td>{OBJECT} shall {ACTION} for {CONDITIONAL}</td>
<td>3</td>
</tr>
<tr>
<td>{OBJECT} shall {ACTION} {RECIPIENT} with {OBJECT}</td>
<td>3</td>
</tr>
<tr>
<td>{CONDITIONAL}, {ACTOR} shall {ACTION} {OBJECT}</td>
<td>3</td>
</tr>
<tr>
<td>{OBJECT} shall {ACTION} to {RECIPIENT}</td>
<td>3</td>
</tr>
<tr>
<td>{ACTOR} is {ACTION} {OBJECT}</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4.1: A sample of templates along with their frequencies

Once the string templates are created from the annotated corpus, it is a straightforward task to convert each template into a regular expression with which to match incoming rules with. Note that adjacent annotation classes need to be combined into one regular expression matching group; at this stage it is unclear which tokens would match the action and which would match the object.

Template: If {Conditional}, {Actor} may {Action} {Object} to {Recipient}

Regular Expression: `If (.*)[ ,] (.*) may (.*) to (.*)$

Figure 4.4: Sample template and the regular expression derived from it
At the end of this stage of training, the model consists of a set of regular expressions to match incoming rules against, as well as a list of rule templates along with their frequency.

4.2.2 Language Models

It is insufficient to merely match each template against input strings, as certain template strings are subsets of other template strings (an example is given in Figure 4.5). For this reason, the model needs some way to score the correctness of the match.

Novel Rule: An agent shall disseminate information.

Candidate Template 1: {ACTOR} shall {ACTION} {OBJECT}
Candidate Extraction 1: ACTOR{An agent} shall ACTION{disseminate} OBJECT{information}

Candidate Template 2: {ACTOR} shall {ACTION}
Candidate Extraction 2: ACTOR{An agent} shall ACTION{disseminate information}

Figure 4.5: A novel rule with two possible template matches

It would be optimal to build a full language model for each annotation class for each template, as this would capture exactly what type of tokens we expect in each blank of each template. For example, if we had seen many annotated training examples, it would be reasonable to conclude that the word “agent” occurs more frequently in the class ACTOR as opposed to the class ACTION. However, given the fact that the training set used for this project is relatively small compared to the domain of English words used in legislation, the data is too sparse to build such a model for this purpose.

As a substitute, the system instead builds a probabilistic model of the parts of speech that are expected for each annotation class. For each sentence in the training corpus, we tag each word with the part of speech for that word. For this purpose, the system uses an already trained n-gram part-of-speech tagger, as implemented in the Python NLTK package [10]. The particular tagger was trained on a human annotated corpus, the Brown Corpus [9], and research in this area shows that automated part-of-speech tagging on English sentences have very high precision for most
sentences. A sample of this construction can be seen in Figure 4.6 and Table 4.2.

ACTOR{Criminal/JJ Justice/NN Agencies/NNS} shall/MD ACTION{disseminate/VB} OBJECT{criminal/JJ justice/NN information/NN}

ACTOR{The/DT central/JJ repository/NN} shall/MD ACTION{share/VB} OBJECT{relevant/JJ documents/NNS}

Figure 4.6: Two annotated training samples, along with the part of speech for each token. JJ is an adjective, NN is a noun, NNS is a plural noun, VB is a verb, MD is a modal auxiliary, DT is a determiner.

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Part of Speech</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTOR</td>
<td>JJ</td>
<td>.333</td>
</tr>
<tr>
<td></td>
<td>DT</td>
<td>.167</td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>.333</td>
</tr>
<tr>
<td></td>
<td>NNS</td>
<td>.167</td>
</tr>
<tr>
<td>ACTION</td>
<td>VB</td>
<td>1.000</td>
</tr>
<tr>
<td>OBJECT</td>
<td>JJ</td>
<td>.400</td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>.400</td>
</tr>
<tr>
<td></td>
<td>NNS</td>
<td>.200</td>
</tr>
</tbody>
</table>

Table 4.2: The probabilities assigned to each part of speech for each annotation class

With this information, for each annotation class in each template, we generate a naive model for the tokens in that class. The system aggregates the part of speech for all of the tokens in each class, and then constructs a probabilistic model for what parts of speech are expected in each class. With this information, we expect that the annotation class models for each template could be different, and that difference is how we prefer one template to another. Referring back to the example in Figure 4.5, matching “disseminate” to the ACTION class and “information” to the OBJECT class in the first template should give a confidence score more than matching “disseminate information” to the ACTION class in the second template, because we expect that the probability of the noun “information” occurring in the ACTION class in the second template is less than the probability that “information” occurs in the OBJECT class.

Note that this naive language model will assign a probability of 0 to any part of speech not found in the training corpus for a given annotation class. Thus, when matching against novel rules,
if a previously unseen part of speech appears, the system will try to assign a probability of 0 to
the match. This is a well-known problem in many areas of machine learning, and the standard
solution is to apply smoothing to the model to account for unseen events [3]. For this system,
simple plus-one smoothing with a small $\gamma$ value serves to smooth the model for each annotation
class. The formula for computing smoothed probabilities can be found in Figure 4.7. A sample
smoothed model is shown in Table 4.3.

$$Pr(POS \mid Class) = \frac{\gamma + count(POS)}{\gamma * N + M}$$

Figure 4.7: Computing the smoothed probability for a part of speech. In this formula, $count(POS)$
is the number of occurrences of the given part of speech, $N$ is the number of different parts of
speech, and $M$ is the total number of tokens encountered in the training set

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Part of Speech</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTOR</td>
<td>JJ</td>
<td>.261</td>
</tr>
<tr>
<td></td>
<td>DT</td>
<td>.134</td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>.261</td>
</tr>
<tr>
<td></td>
<td>NNS</td>
<td>.134</td>
</tr>
<tr>
<td></td>
<td>VB</td>
<td>.006</td>
</tr>
<tr>
<td></td>
<td>MD</td>
<td>.006</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>.006</td>
</tr>
<tr>
<td>ACTION</td>
<td>VB</td>
<td>.532</td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>.133</td>
</tr>
<tr>
<td></td>
<td>JJ</td>
<td>.013</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>.013</td>
</tr>
<tr>
<td>OBJECT</td>
<td>JJ</td>
<td>.299</td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>.299</td>
</tr>
<tr>
<td></td>
<td>NNS</td>
<td>.153</td>
</tr>
<tr>
<td></td>
<td>VB</td>
<td>.007</td>
</tr>
<tr>
<td></td>
<td>MD</td>
<td>.007</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>.007</td>
</tr>
</tbody>
</table>

Table 4.3: The probabilities assigned to each part of speech for each annotation class, after smooth-
ing is applied with $\gamma = .05$

Another example where this probabilistic model aid in template matching is shown in Figure 4.8. For the first candidate, the system tries to extract the string “Criminal history information”
as an ACTOR, while for the second we extract the string “Criminal history information” as an OBJECT.

In our training data, we have seen other examples of the template “\{OBJECT\} shall \{ACTION\} if \{CONDITIONAL\}”, and in each of those cases, the tokens annotated as ACTION were of the form “be \textit{verb}”. This is reflected in the model, as we give equal weight to the parts of speech VB (Verb) and VBN (Verb, past participle). However, in the training cases where the ACTOR comes before shall, we do not see the construction “be \textit{verb}”, and instead just see examples of “\textit{verb}”. Again, this is reflected in the model for ACTION in the first template. After annotating the input rule with the relevant parts of speech, it becomes clear that candidate 2 is a better match, as the extraction “be disseminated” has a better probability of coming from candidate 2.

**Novel Rule:** Criminal history information shall be disseminated if there is probable cause.

**Candidate Template 1:** ACTOR shall ACTION if CONDITIONAL

**Candidate Template 2:** OBJECT shall ACTION if CONDITIONAL

**Candidate 1 ACTION model:**
POS: VB P: 1.000

**Candidate 2 ACTION model:**
POS: VB P: 0.500
POS: VBN P: 0.500

**Figure 4.8:** Example where the model for ACTION in two different templates are different

### 4.2.3 Training Outcome

After processing the training corpus as detailed in this section, the trained system consists of 3 parts:

1. A set of regular expressions to match incoming novel rules against.

2. A list of seen templates along with the annotated rules associated with each template.
3. For each template, a map that takes as input an annotation class and a part of speech, which
returns the probability of that part of speech being in that annotation class for this template.

4.3 Annotating Novel Rules

This section will describe how, after being trained, the system can annotate novel inputs. The input
to the system is a plain English string. The output of the system is a list of potential matches, each
of which consists of:

1. A confidence score for the given match

2. The input string with annotations attached, where the potential annotations are ACTOR,
   ACTION, OBJECT, RECIPIENT, and CONDITIONAL.

3. A list of similar rules encountered while the system was trained. If these rules have been
   already translated by the user to the target language, these similar rules will prompt the user
   as to what type of structure in the formal language is appropriate for the translation.

A sample synthetic unannotated rule can be seen in in Figure 4.9. This example will be used
in the rest of this section. The trained annotator used for the remainder of the section is trained on
the entire annotated corpus.

As long as the agent lawfully collected the information, the criminal justice agency, if given a request in writing beforehand, may disseminate information about an individual to another agency.

Figure 4.9: Unannotated rule for the following section

The overall pseudocode of the annotation process can be found in Figure 4.10. The details of
this process will be explained in the following subsections.
function annotate(input_rule):
    conditional_annotations, reduced_input_rule =
        annotate_conditional_heuristics(input_rule)
    candidates = []
    for template in templates:
        if not template.match(reduced_input_rule):
            continue
        annotated_rule = template.annotate(reduced_input_rule)
        score = template.score(reduced_input_rule)
        annotated_rule.merge(conditional_annotations)
        candidates.append((score, annotated_rule))
    return sort(candidates)

Figure 4.10: Pseudocode for evaluating an unknown input rule

4.3.1 Annotating Conditional Clauses

Conditional clauses are particularly problematic for the template matching system detailed above.

1. The common words or phrases that signal that a condition is following are very diverse (e.g.
   "if", "unless", "except", "as long as").

2. The location of the conditional phrase in the rule can vary greatly without changing the
   meaning. For example, "If there is probable cause, the agent shall share" has an identical
   meaning to "The agent shall, if there is probable cause, share" as well as "The agent shall
   share if there is probable cause".

Because of the two reasons listed above, the number of templates that contain conditionals
are very sparsely distributed (see Table 4.4). Because of this, for any novel rule that comes into
the system, if it has conditional clauses, it is relatively unlikely that we have seen a rule in our training set that has the exact same structure, and thus the trained system will not successfully find the correct template with which to match the incoming rule. In the example given in Figure 4.9, we would have had to seen a template in our training set of “As long as {CONDITIONAL}, {ACTOR}, if {CONDITIONAL}, may {ACTION} {OBJECT} to {RECIPIENT}”, which is highly unlikely.

Referring back to the central hypothesis of this work (that there are only a handful of ways rules are expressed), the caveat is that conditionals can appear in so many different places in a rule, with so many different phrases prompting a conditional, that the performance of the system is greatly improved if the conditional is removed before the rest of the process.

<table>
<thead>
<tr>
<th>Template</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>notwithstanding {CONDITIONAL}, {ACTOR} may {ACTION} {OBJECT} to {RECIPIENT} in accordance with {CONDITIONAL}</td>
<td>1</td>
</tr>
<tr>
<td>except {CONDITIONAL}, {OBJECT} shall {ACTION} in accordance with {CONDITIONAL}</td>
<td>1</td>
</tr>
<tr>
<td>{ACTOR} shall except for {CONDITIONAL}, {ACTION} {OBJECT}</td>
<td>2</td>
</tr>
<tr>
<td>in {CONDITIONAL}, {ACTOR} shall {ACTION} {OBJECT}</td>
<td>3</td>
</tr>
<tr>
<td>pursuant to {CONDITIONAL}, {ACTOR} may {ACTION} {OBJECT} to {RECIPIENT} for the purpose of {CONDITIONAL}</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.4: A list of templates with conditionals and their frequencies, as seen in the corpus annotated in Chapter 3. Note that the occurrence of each of these is very small.

In this system, this problem is solved with a heuristic that seeks to extract conditionals from input rules before running it through the rest of the system. There is a list of hard-coded phrases (see Table 4.5) that signal with high probability that the tokens following it should be annotated as a conditional. This list of indicators was cultivated from the annotated corpus for this project by hand.

The heuristic aims to extract conditionals from three different locations in input rules:
unless; pursuant to; whenever; to the extent that; until; in regard to; in connection with; before; whether; when
except; upon; without; with the exception of; subject to; in support of within; in addition to; prior to; within
if; notwithstanding; except for; for the purpose of; during; whether or not; in order to; throughout and upon; where
in accordance with; after; as long as; for the purposes of; in such a manner; at the time of; where there is; at least; but

Table 4.5: List of phrases that signal an upcoming conditional phrase

1. If the rule begins with a conditional indicator, extract up to the next comma (e.g. “Unless ..., the agent shall...”).

2. If the word token preceding the conditional indicator is a modal auxiliary (“shall”, “may”, “will”, etc.), extract up to the next comma (e.g. “The agent shall, pursuant to Title I, disseminate...”).

3. If there are no modal auxiliaries after the conditional indicator, extract to the end of the rule (e.g. “The agent shall disseminate information if ....”).

In practice, these heuristics have very high accuracy in tagging conditional phrases that are part of the input (see Chapter 5), and renders the remainder of the input easier to process, as removing conditional phrases dramatically shrinks the set of potential templates that could be matched. After each of these heuristics are applied, the conditional indicator tokens as well as the entire conditional phrase are no longer considered for the rest of the annotation process (see Figure 4.11). When the output is produced, these extracted conditionals are added back into the output for every template match.

44
As long as the agent lawfully collected the information, the criminal justice agency, if given a request in writing beforehand, may disseminate information about an individual to another agency

Heuristic 1:
"the agent lawfully collected the information" annotated as CONDITIONAL
"As long as the agent lawfully collected the information," hidden from the rule.

Heuristic 2:
"given a request in writing beforehand" annotated as CONDITIONAL
"...if given a request in writing beforehand," hidden from the rule.

Remaining rule after heuristics:

the criminal justice agency may disseminate information about an individual to another agency

Figure 4.11: Unannotated rule with heuristics extracting conditionals

4.3.2 Template Matching

After conditionals have been extracted, the remaining string is matched against all of the regular expressions generated during training. For each regular expression, if a match is found, the template that generated the regular expression will dictate which annotation class each matched group belongs to (see Figure 4.12). If no templates are matched at all, then the input rule is completely novel to the system, and no extractions are possible.
the criminal justice agency may disseminate information about an individual to another agency

Template 1: \{ACTOR\} may \{ACTION\} \{OBJECT\}
Extracted elements:
  \begin{itemize}
    \item ACTOR = the criminal justice agency
    \item ACTION/OBJECT = disseminate information about an individual to another agency
  \end{itemize}

Template 2: \{ACTOR\} may \{ACTION\} \{OBJECT\} to \{RECIPIENT\}
Extracted elements:
  \begin{itemize}
    \item ACTOR = the criminal justice agency
    \item ACTION/OBJECT = dissemination information about an individual
    \item RECIPIENT = another agency
  \end{itemize}

Template 3: \{OBJECT\} may \{ACTION\}
Extracted elements:
  \begin{itemize}
    \item OBJECT = the criminal justice agency
    \item ACTION = disseminate information about an individual to another agency
  \end{itemize}

Figure 4.12: Matching the input rule against seen templates

4.3.3 Extraction Token Classification

Note that in Figure 4.12, since some templates have two annotation classes adjacent to each other, it is unclear which tokens in the extraction belong to which class. For example, in the match for template 2, the system deduced that "disseminate information about an individual" contains an ACTION followed by an OBJECT. However, the template does not tell us which tokens in the match belong to which class.

This problem is solved by the language models derived in the training portion. For each template, the system has a function that maps (annotation class, part of speech) into the probability that that part of speech belongs to that annotation class. A snippet of that function is shown in Table 4.6.

Each token of the input rule is tagged with its part of speech. The algorithm considers splitting the tokens in the match at each index, with the tokens on the left of the split point being in the left annotation class, and the tokens on the right of the split point being in the right annotation class.
Table 4.6: The probability function for (annotation class, POS) for the template \{ACTION\} \{OBJECT\} to \{RECIPIENT\}

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Part of Speech</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION</td>
<td>VB</td>
<td>.667</td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>.071</td>
</tr>
<tr>
<td></td>
<td>IN</td>
<td>.003</td>
</tr>
<tr>
<td></td>
<td>DT</td>
<td>.003</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>OBJECT</td>
<td>VB</td>
<td>.012</td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>.358</td>
</tr>
<tr>
<td></td>
<td>IN</td>
<td>.127</td>
</tr>
<tr>
<td></td>
<td>DT</td>
<td>.093</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Using the model, the system computes a probability for each potential split point (see Figure 4.13), and considers the split with the highest probability to be the most likely partition.

disseminate\textsc{}/\textsc{VB} information\textsc{}/\textsc{NN} about\textsc{}/\textsc{IN} an\textsc{}/\textsc{DT} individual\textsc{}/\textsc{NN}

\[
\Pr(\text{ACTION\{disseminate\}} \ \text{OBJECT\{information about an individual\}}) \\
= \Pr(\text{ACTION, VB}) \times \Pr(\text{OBJECT, NN}) \times \Pr(\text{OBJECT, IN}) \times \Pr(\text{OBJECT, DT}) \times \Pr(\text{OBJECT, NN}) \\
= .667 \times .358 \times .127 \times .093 \times .358 \\
= 1.0 \times 10^{-3}
\]

\[
\Pr(\text{ACTION\{disseminate information\}} \ \text{OBJECT\{about an individual\}}) \\
= .667 \times .071 \times .127 \times .093 \times .358 \\
= 2.0 \times 10^{-4}
\]

\[
\Pr(\text{ACTION\{disseminate information about\}} \ \text{OBJECT\{an individual\}}) \\
= .667 \times .071 \times .003 \times .093 \times .358 \\
= 4.7 \times 10^{-6}
\]

\[
\Pr(\text{ACTION\{disseminate information about an\}} \ \text{OBJECT\{individual\}}) \\
= .667 \times .071 \times .003 \times .003 \times .358 \\
= 1.5 \times 10^{-7}
\]

Figure 4.13: Calculating the probability for each split point

As seen in Figure 4.13, the split of “ACTION\{disseminate\} OBJECT\{information about an individual\}” has the highest probability. In this way, for each matching template, the system will
be able to annotate the input rule with annotation classes (see Figure 4.14). Note that there is no longer any ambiguity between ACTION and OBJECT classes in templates 1 and 2.

the criminal justice agency may disseminate information about an individual to another agency

Template 1: {ACTOR} may {ACTION} {OBJECT}

Extracted elements:
ACTOR = the criminal justice agency
ACTION = disseminate
OBJECT = information about an individual to another agency

Template 2: {ACTOR} may {ACTION} {OBJECT} to {RECIPIENT}

Extracted elements:
ACTOR = the criminal justice agency
ACTION = disseminate
OBJECT = information about an individual
RECIPIENT = another agency

Template 3: {OBJECT} may {ACTION}

Extracted elements:
OBJECT = the criminal justice agency
ACTION = disseminate information about an individual to another agency

Figure 4.14: Extracting classes after finding optimal partitions
4.3.4 Match Scoring

At this point, for our input rule, the system has a list of matching templates, and for each template has a list of extractions. In order for the user to make sense of this output, it is useful to assign a confidence score to each of the matches and then sort the list of matches, so the match the system has the most confidence in is presented to the user first.

For scoring each match, the system considers the probability $Pr(T_i|R)$, where $T_i$ is the $i$th matching template, $R$ is the input rule, and $w_i$ is the $i$th token of the input rule. The system uses a Naive Bayes probabilistic model to calculate this probability.

$$Pr(T|R) = \frac{Pr(T_i) * Pr(R|T_i)}{Pr(R)}$$

$$\propto Pr(T_i) * Pr(R|T_i)$$

$$= Pr(T_i) * \prod_{j=1}^{ |R| } Pr(w_i|T_i)$$

$$\log(Pr(T|R)) \propto \log(Pr(T_i)) + \sum_{j=1}^{ |R| } \log(Pr(w_i|T_i))$$

In this way, the log of the probability is a method to score each individual match. $Pr(T_i)$ is calculated using the maximum likelihood estimate (MLE), by taking $\frac{\text{count}(T_i)}{N}$, where $N$ is the total number of different templates encountered in the training set. $Pr(w_i|T_i)$ is assigned to be the probability of the part of speech of $w_i$ under the annotation assigned by $T_i$ to $w_i$. These probabilities are the smoothed (annotation, part of speech) probabilities calculated during the training phase of the system.

Note that it is assumed that $Pr(w_i|T_i) = 1$ if $w_i$ is not annotated by the template. This is the case for words like “may” in all three templates in the example, and “to” in template 2. In a sense, this promotes template 2 as a better match to the original string, as $Pr(to|T_2) = 1$, while $Pr(to|T_1) < 1$.
and $Pr(to|T_3) < 1$. This is sensible, as matching more tokens to the template in question is a signal that it is a better match. The probabilities for template 2 are demonstrated in Figure 4.15; the scores for the other two templates are derived in an identical manner.

$$\log(Pr(T_2|R)) \propto \log(Pr(T_2)) + \sum_{j=1}^{R} \log(Pr(w_j|T_2))$$

$$= \log\left(\frac{\text{count}(T_2)}{N}\right) + \log(Pr(\text{the}|T_2)) + \log(Pr(\text{criminal}|T_2)) +$$

$$... + \log(Pr(\text{another}|T_2)) + \log(Pr(\text{agency}|T_2))$$

$$= \log(\frac{\text{count}(T_2)}{N}) + \log(Pr(\text{ACTOR,DT}|T_2)) + \log(Pr(\text{ACTOR,JJ}|T_2)) +$$

$$... + \log(Pr(\text{RECIPIENT,JJ}|T_2)) + \log(Pr(\text{RECIPIENT,NN}|T_2))$$

$$= -3.354 - 1.482 - 1.817 ... - 2.731 - 1.128$$

$$= -22.5423$$

$$\log(Pr(T_1|R)) = -23.6835$$

$$\log(Pr(T_3|R)) = -53.8573$$

Figure 4.15: Calculating the scores for matching each template against the novel rule

Note that, as expected, template 1 and template 2 have scores that are relatively close together, with template 2 getting the slight advantage (probably due to $Pr(to|T_2) = 1$, as noted above). Template 3 is a very bad match for this input rule due to the fact that many tokens that do not belong in an ACTION are being annotated as one; the significantly lower score reflects this.

4.3.5 Summary

Finally, after scoring, the output is ready for the user. In the case of this example, the final output is shown in Figure 4.16. The similar rules are simply sample rules from the input training corpus that have the same template as the input rule.

The input used as a sample for this chapter was synthetic, as it was convenient to demonstrate all of the important features of the system. A genuine rule from IIR 28 CFR Part 23 can be seen in Figure 4.17, along with output when run through an annotator trained with the annotated corpus.
Input: As long as the agent lawfully collected the information, the criminal justice agency, if given a request in writing beforehand, may disseminate information about an individual to another agency

Template 2: {ACTOR} may {ACTION} {OBJECT} to {RECIPIENT}
Score: -22.5423
Extracted elements:
ACTOR = the criminal justice agency
ACTION = disseminate
OBJECT = information about an individual
RECIPIENT = another agency
CONDITIONAL = the agent lawfully collected the information
CONDITIONAL = given a request in writing beforehand
Similar rules:
A criminal justice agency may not disseminate CHRI to another criminal justice agency until the disseminating agency has requested and received from the central repository verification that the information to be disseminated is complete, accurate, and current
...

Template 1: {ACTOR} may {ACTION} {OBJECT}
Score: -23.6835
Extracted elements:
ACTOR = the criminal justice agency
ACTION = disseminate
OBJECT = information about an individual to another agency
CONDITIONAL = the agent lawfully collected the information
CONDITIONAL = given a request in writing beforehand
Similar rules:
A project may authorize and utilize remote (off-premises) system data bases to the extent that they comply with these security requirements
...

Template 3: {OBJECT} may {ACTION}
Score: -53.8573
OBJECT = the criminal justice agency
ACTION = disseminate information about an individual to another agency
CONDITIONAL = the agent lawfully collected the information
CONDITIONAL = given a request in writing beforehand
Similar rules:
Unless otherwise authorized by this section, information about the juvenile record may not be released when the request for information is related to an application for employment, license, bonding, or any civil right or privilege
...

Figure 4.16: The final output of the system given the sample input
Input: A project shall collect and maintain criminal intelligence information concerning an individual only if there is reasonable suspicion that the individual is involved in criminal conduct or activity and the information is relevant to that criminal conduct or activity.

Template 1: {ACTOR} shall {ACTION} {OBJECT} if {CONDITIONAL}
Score: -77.479
Extracted elements:
ACTOR = A project
ACTION = collect and maintain
OBJECT = criminal intelligence information concerning an individual
CONDITIONAL = there is reasonable suspicion that the individual is involved in criminal conduct or activity and the information is relevant to that criminal conduct or activity

Similar Rules:
agencies or individuals granted access under clause (c) shall be eligible to receive criminal offender record information obtained through interstate systems if the board determines that such information is necessary for the performance of the actions or duties sustaining the public interest with respect to such agencies or individuals ...

Template 2: {ACTOR} shall {ACTION} {OBJECT}
Score: -89.821
Extracted elements:
ACTOR = A project
ACTION = collect and maintain
OBJECT = criminal intelligence information concerning an individual if there is reasonable suspicion that the individual is involved in criminal conduct or activity and the information is relevant to that criminal conduct or activity

Similar Rules:
...

Template 3: {ACTOR} shall {ACTION} {OBJECT} to {RECIPIENT}
Score: -128.359415
ACTOR = A project
ACTION = collect and maintain
OBJECT = criminal intelligence information concerning an individual if there is reasonable suspicion that the individual is involved in criminal conduct or activity and the information is relevant
RECIPIENT = that criminal conduct or activity

Similar Rules:
...

Figure 4.17: The final output of the system given the rule from IIR 28 CFR Part 23
Chapter 5

Evaluation

This chapter details the process used to evaluate the system developed in the preceding chapters. It explains the experimental setup, reveals the results of the experiment, gives explanations for situations in which the system performs poorly, and proposes potential solutions for the next iteration of the system.

5.1 Experimental Setup

As detailed in Chapter 3, 315 rules were annotated by hand so that each rule was divided into portions detailing the actor, action, object, recipient, and relevant conditions. For evaluation, cross validation is used with 10 partitions, so that for each iteration of cross validation 90% of the annotated corpus is used as training data while 10% of the annotated corpus is used as an evaluation set.

The annotator is trained with the 90% portion as detailed in Chapter 4. The remaining 10% portion is given as input to the annotator, and the annotations applied by the trained annotator are compared against the hand-applied annotations. All of the scores in this chapter are the average over all 10 trials.

There are several ways to measure the effectiveness of the system, given the viewpoint that the original aim is to help a human translator translate from natural language to a formal language.
The system should efficiently:

1. Extract the appropriate conditional phrases from each rule using the heuristics defined in Section 4.3.1.

2. Given that the conditionals are extracted, try to match the novel rule against rule templates that the system has seen so far.

3. Accurately annotate the rule on a token-by-token basis, independent of the correctness of the template match. That is, even if the wrong template is matched, it still may be the case that a large number of tokens are correctly annotated if the matched template and the real template only differ slightly.

Each of the three evaluation metrics listed above will be analyzed below in detail.

### 5.2 Evaluation of Conditional Extraction

As stated in Section 4.3.1, a series of heuristics are applied to each incoming rule to deal with the problem of sparseness in rules with conditionals. It is possible to evaluate these heuristics by evaluating the precision and recall of tokens tagged as belonging to a conditional versus the conditionals annotated by hand. Since the three heuristics listed in Section 4.3.1 are mutually exclusive in their applicability, it is possible to evaluate each of them individually. The precision, recall, and f-measure of each of the three techniques are given in Table 5.1.

<table>
<thead>
<tr>
<th></th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head Conditional Extraction</td>
<td>.209</td>
<td>.974</td>
<td>.324</td>
</tr>
<tr>
<td>Inline Conditional Extraction</td>
<td>.046</td>
<td>1.000</td>
<td>.085</td>
</tr>
<tr>
<td>Tail Conditional Extraction</td>
<td>.207</td>
<td>.617</td>
<td>.289</td>
</tr>
<tr>
<td>Combined</td>
<td>.462</td>
<td>.756</td>
<td>.556</td>
</tr>
</tbody>
</table>

Table 5.1: Evaluation scores for the heuristic-based conditional tagging mechanism
5.2.1 Analysis of Head Conditional Extraction

This heuristic performed with very high accuracy, as there is very little variation in conditionals expressed in this manner. There is only one general case in which this heuristic fails occasionally.

In `CONDITIONAL{assessing good cause} ACTOR{the court} shall ACTION{weigh} OBJECT{the public interest and the need for disclosure against the injury to the patient, to the physician-patient relationship, and to the treatment services}`.

Figure 5.1: Rule where head conditional extraction fails, from US Code Title 42 Chapter 6A Subchapter III-A Part D 290dd2

In Figure 5.1, there is no comma at the end of the conditional as there is in the vast majority of other similar rules. Instead, the heuristic extracts all tokens up to “injury to the patient” as part of the conditional, which is incorrect. A similar problem exists when there is a comma-separated list in the conditional; in this case, the heuristic will only extract the first part of the conditional, and leave the rest of the list. An improved heuristic could patch this flaw by trying to find modal auxiliaries in the extraction (the “shall” that is extracted as part of the conditional). However, given the high precision of this heuristic as-is, this seems to be a minor issue.

5.2.2 Analysis of Inline Conditional Extraction

This rule performed with perfect accuracy, but very low recall. This is mostly an artifact of the fact that in real rules, expressing conditionals in this manner is uncommon (out of 315 rules, only 19 had a conditional expressed this way). In addition, the high standard of applying this heuristic (a preceding modal auxiliary and a trailing comma) lead to very accurate matches.

5.2.3 Analysis of Tail Conditional Extraction

This heuristic is the one that leads to the most incorrect annotations, but is still effective for extracting conditional phrases from rules. The primary failure situation is displayed in Figure 5.2.
ACTOR{The department} shall ACTION{collect} OBJECT{information concerning the number and nature of offenses known to have been committed in this state and of the legal steps taken in connection with these offenses, such other information that is useful in the study of crime and in the administration of criminal justice and all other information deemed necessary to operate the statewide uniform crime reporting program and to cooperate with the federal government uniform crime reporting program}.

Figure 5.2: Rule where tail conditional extraction fails, from Arizona 41-1750.

The heuristic finds the string literal “in connection with”, which leads it to extract everything that follows as a conditional annotation. However, in this case, “in connection with” modifies the object being collected by the department, and does not govern the conditions when the department shall collect.

Most failures of this heuristic are caused by this phenomenon. One approach to solving this problem is to generate a full parse tree of the original rule as part of the annotation procedure, and then figure out that this particular “in connection with” is not referring to the entire rule but to a smaller subset of the rule. However, doing so would dramatically increase the amount of NLP machinery that is present in the annotator, and would require training a parse tree generator for this corpus.

5.3 Evaluation of Template Matching

Corresponding to Section 4.3.2, after the conditional heuristics are applied, the system tries to match the set of all templates that the trained annotator has seen against the novel rule in question. Each successfully matched template will produce a score (as described in Section 4.3.4), which is then used to rank all of the matches in terms of suitability. Two metrics are of interest in this case:

1. Given a novel rule, what is the likelihood that its rule template is seen in the training data? If it is not, then the best the annotator can hope to do is find a near match (e.g. “ACTOR shall ACTION OBJECT to RECIPIENT” is close to “ACTOR shall ACTION OBJECT”), as the annotator has no mechanism to make brand new templates after it has been trained.
2. Assuming that we have seen the novel rule’s template in the training set, how often is that template given the best score by the annotator? Or, more generally, how often is the correct template in the top $N$-scored templates by the annotator?

The results for the above two questions are shown in Table 5.2.

<table>
<thead>
<tr>
<th>Correct Match in Top-N</th>
<th>Proportion of Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.432</td>
</tr>
<tr>
<td>2</td>
<td>.502</td>
</tr>
<tr>
<td>3</td>
<td>.534</td>
</tr>
<tr>
<td>5</td>
<td>.585</td>
</tr>
<tr>
<td>10</td>
<td>.664</td>
</tr>
<tr>
<td>$\infty$</td>
<td>.692</td>
</tr>
</tbody>
</table>

Table 5.2: Results for template matching

5.3.1 Analysis of Template Matching Failures

This section attempts to analyze the roughly 30% of novel input rules where the correct template is not found in the trained annotator. Note that this will be the case for any rule in the annotated training set that does not share a template with any other rule. If this is the case, when the given rule is being evaluated, there are certainly no rules with the same template in the trained annotator, so template matching will fail. A small sample of these rule templates can be seen in Table 5.3.

{OBJECT} shall {ACTION} to {RECIPIENT} and shall {ACTION} by {ACTOR}
{OBJECT} shall {ACTION} by {ACTOR} to {RECIPIENT}
{OBJECT} may {ACTION} by {ACTOR}
{CONDITIONAL}, {ACTOR} may {ACTION} {OBJECT} with {RECIPIENT}
{ACTOR} must {ACTION} {OBJECT} to {CONDITIONAL} and {ACTION} {OBJECT}
{ACTOR} {ACTION} {OBJECT} with {RECIPIENT}
{ACTOR} shall {ACTION} from {RECIPIENT} {OBJECT}
{ACTOR} shall {ACTION} {OBJECT} and {ACTION} to {RECIPIENT} {OBJECT}

Table 5.3: Rule templates that are unique in the training set

This sample demonstrates some of the flaws of the implemented system.
1. The system is not optimized to deal with rules that have an “and” or an “or”. Rules with conjunctions have the same sparseness problem as rules with conditionals, in that there is a huge variation in the annotations that can be on either side of the conjunction, and any generated templates will have little corroboration between them. For example, in the first template cited in Table 5.3, it is possible to rewrite the underlying rule as “{OBJECT} shall {ACTION} to {RECIPIENT}” with a second rule of “{OBJECT} shall {ACTION} by {ACTOR}”, and both of these derived rules would be a separate template match. However, such a rewriting rule would have to be highly specialized, and would also require a parse tree of the rule so as to not pick up “and” and “or” occurrences inside an actor, object, or conditional instead of the main rule.

2. To a lesser degree, there is a similar problem with recipients as there is with conditionals, in that they can be attached to a rule in different places and with different words that signify a recipient (“with”, “to”). A similar heuristic as the conditional extraction heuristic may prove useful, though given the significantly less recipients in the annotated rules, this would not provide as much of an improvement in performance.

The underlying hypothesis of this entire system is that there are relatively few rules templates that govern the way rules are expressed in this domain. By this metric, it is shown that when trained on a few hundred rules and evaluating a few dozen rules, about 70% of the rules will have had templates that were seen before. In the above section, steps were proposed that could be used to increase this coverage even further. So, this is evidence that the hypothesis is true; there are only a limited number of rule structures that are generally used to express rules in this domain.

5.3.2 Analysis of Template Scoring Failures

This section attempts to analyze the roughly 26% of novel input rules where the correct template has been seen by the annotator, but another template has been given a higher score over it.
Template Occurring Too Infrequently

As detailed in Section 4.3.4, the log-likelihood of a match is calculated using a Bayesian model, where the likelihood of the template in the training data is taken into account along with the likelihood of each of the annotations associated with each token. However, the set of annotated templates is skewed to a few very frequently occurring templates ("{ACTOR} shall {ACTION} {OBJECT}", "{ACTOR} may {ACTION} {OBJECT}"). So, the scoring mechanism will tend to favor these template matches more highly over other template matches, even if the token probabilities favor the latter.

The information pertaining to dissemination required to be maintained shall be retained for a period of not less than one year.

Template: {1} shall {2} {3}
Score: -64.532059 Template prior: -1.183066 Token probability: -63.348993

Template: {3} shall {2} for {5}
Score: -65.421868 Template prior: -4.550362 Token probability: -60.871506

Figure 5.3: Two template matches for a rule from Washington RCW 10.97.050

In Figure 5.3, the correct template is the second one. The token probabilities are higher for the second template, however, because it occurs so infrequently, the first template is still favored over it.

A solution to this problem might be to develop a more sophisticated smoothing mechanism for the template probabilities, so that extremely frequent templates did not dominate the scoring.

Insufficient Template Occurrences for Language Modeling

As detailed in Section 4.2.2, for each unique template, the annotator attempts to build a language model based on the parts-of-speech for each annotation class. This is an effective method for frequently occurring templates, but is less effective for training templates seen infrequently, as the model will be extremely overfitted for the few training examples it has seen.
Use of CHRI disseminated to noncriminal justice agencies shall be limited to the purpose for which it was given.

Template: {OBJECT} shall {ACTION} to {RECIPIENT}
Score: -44.365630 Template prior: -3.699537 Token probability: -40.666093

Template: {OBJECT} shall {ACTION} to {CONDITIONAL}
Score: -90.248831 Template prior: -5.645447 Token probability: -84.603384

Figure 5.4: Two template matches for a rule from Kentucky 502 KAR 30:060

In Figure 5.4, the correct match has a much lower score than the first match. In the training set, the second template occurs exactly twice; once in this rule, and once in the rule displayed in Figure 5.5.

Figure 5.5: The one rule instance that derived the correctly matching template

As explained in Section 4.2.2, plus-one smoothing is used to create the language model for this specific template. Since there is only one training instance for this template, the probability models for this template are highly overfitted for this specific template. For example, since no verbs occur in the OBJECT class for the training instance, the verb “disseminated” in the novel instance will have a very low probability for matching the OBJECT model of the template.

One way to potentially resolve this problem is to use a mixture model for the probabilities for each part of speech in each annotation class. That is, the annotator could create a probability function for each annotation class independent of template. Then, for each template, scale the probability to be a function of the global probability function and the individual template’s probability function:

\[ P'(class|template, pos) = \lambda_1 P(class|template, pos) + \lambda_2 P(class|pos) \]
Thus, even for infrequently occurring templates, the language models for that template are not drastically overfitted.

5.3.3 Evaluation of Template Matching Without Conditional Heuristics

As a brief aside, the justification for the conditional heuristics is that the template set is too sparse for practical matching to occur. In other words, if we disable the conditional heuristics described in Section 4.3.1, the figures in Table 5.2 should decrease. This is in fact the case, and is shown in Table 5.4

<table>
<thead>
<tr>
<th>Correct Match in Top-N</th>
<th>Proportion of Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.289</td>
</tr>
<tr>
<td>2</td>
<td>.337</td>
</tr>
<tr>
<td>3</td>
<td>.346</td>
</tr>
<tr>
<td>5</td>
<td>.407</td>
</tr>
<tr>
<td>10</td>
<td>.460</td>
</tr>
<tr>
<td>∞</td>
<td>.470</td>
</tr>
</tbody>
</table>

Table 5.4: Results for template matching when conditional extraction heuristics are disabled

5.4 Evaluation of Per-Token Annotations

Finally, we consider the annotations on a per-token instance. That is, independent of template matching and the conditional extraction heuristics, we can evaluate for each rule which tokens were tagged correctly by the annotator and which were not. Note that a correct template match does not guarantee that all of the tokens will be annotated correctly, due to potential errors in the splitting algorithm detailed in Section 4.3.3. Also note that an incorrect template match may still induce many correct annotations. The results are shown in Table 5.5.

Note that the system performs well on actors, actions, and objects. The likely explanation for this is that even if the wrong template is picked, with reasonable probability the incorrect template will still extract the actor, action, and object correctly.
<table>
<thead>
<tr>
<th></th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>.850</td>
<td>.604</td>
<td>.702</td>
</tr>
<tr>
<td>Action</td>
<td>.766</td>
<td>.765</td>
<td>.745</td>
</tr>
<tr>
<td>Object</td>
<td>.788</td>
<td>.678</td>
<td>.726</td>
</tr>
<tr>
<td>Recipient</td>
<td>.081</td>
<td>.413</td>
<td>.124</td>
</tr>
<tr>
<td>Conditional</td>
<td>.476</td>
<td>.749</td>
<td>.563</td>
</tr>
<tr>
<td>Total</td>
<td>.668</td>
<td>.667</td>
<td>.668</td>
</tr>
</tbody>
</table>

Table 5.5: Results for individual token annotations

The extraction process for recipients is poor. It is a conflation of the two issues in Section 5.3.2; the templates that have a recipient listed occur relatively infrequently so they will be selected less often, and the part-of-speech models for these templates are overfitted to the few training examples seen.

Finally, the extractions for conditionals is reasonable, though could use significant improvement. The issues revolving around conditional extraction were explored in depth in Section 5.2.
Chapter 6

Conclusion

6.1 Future Work

6.1.1 Application to Other Domains

Part of the reason that this system works in this domain is that real world legislation of this nature is very highly structured. There are only a handful of ideas that are possibly expressed, and generally only a handful of things change between different inputs. This is why such a simplistic model of language is suitable for this domain.

It would be worth investigating if any other domains of language fall into this category. That is, even though the input theoretically spans all natural language, in practice there is very little variation in overall structure between inputs. Potential domains for testing this include other domains of regulation and law, as well as documents relating to healthcare.

6.1.2 Expansion of the Data Model Used

As stated in the chapter on data, all of the input was reduced to the lowest common denominator; that of a series of sentences that all act independently of each other. It was already pointed out that this model is sufficient for parsing and translating a majority of stand-alone rules, but is unsuitable for some rules. It would be worth investigating if a more robust data model would be helpful, and
if implemented, if it would increase the applicability of this technique to other domains.

6.2 Summary

In this work, I have implemented a system that helps people translate natural language rules into equivalent rules written in a formal language. Given the myriad of possible target languages and the difficulty in translating entities into particular elements of the formal language vocabulary, the system merely extracts the relevant portions of each rule for the user, and also gives the user a list of similar rules that the system was trained on. In this way, the user can use a previously translated rule to minimize the amount of future translation work they can do; the user can merely copy the translated version of the old rule and plug in the new entities.

In order to accomplish the above goals, I have also annotated a substantial body of real-world legislation in order to train such a system. In doing so, it highlighted the particular structure of the existing legislation that this system tries to exploit, as well as demonstrates the homogeneous nature of the language in this domain.
Bibliography


