Factoring Statutory Reasoning as Language Understanding Challenges

Nils Holzenberger and **Benjamin Van Durme**
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, Maryland, USA
nilsh@jhu.edu vandurme@cs.jhu.edu

Abstract

Statutory reasoning is the task of determining whether a legal statute, stated in natural language, applies to the text description of a case. Prior work introduced a resource that approached statutory reasoning as a monolithic textual entailment problem, with neural baselines performing nearly at-chance. To address this challenge, we decompose statutory reasoning into four types of language-understanding challenge problems, through the introduction of concepts and structure found in Prolog programs. Augmenting an existing benchmark, we provide annotations for the four tasks, and baselines for three of them. Models for statutory reasoning are shown to benefit from the additional structure, improving on prior baselines. Further, the decomposition into subtasks facilitates finer-grained model diagnostics and clearer incremental progress.

1 Introduction

As more data becomes available, Natural Language Processing (NLP) techniques are increasingly being applied to the legal domain, including for the prediction of case outcomes (Xiao et al., 2018; Vacek et al., 2019; Chalkidis et al., 2019a). In the US, cases are decided based on previous case outcomes, but also on the legal statutes compiled in the US code. For our purposes, a *case* is a set of facts described in natural language, as in Figure 1, in blue. The US code is a set of documents called *statutes*, themselves decomposed into *subsections*. Taken together, subsections can be viewed as a body of interdependent rules specified in natural language, prescribing how case outcomes are to be determined. *Statutory reasoning* is the task of determining whether a given subsection of a statute applies to a given case, where both are expressed in natural language. Subsections are implicitly framed as predicates, which may be true or false of a given case. Holzenberger et al. (2020) introduced SARA, a benchmark for the task of statutory reasoning, as well as two different approaches to solving this problem. First, a manually-crafted symbolic reasoner based on Prolog is shown to perfectly solve the task, at the expense of experts writing the Prolog code and translating the natural language case descriptions into Prolog-understandable facts. The second approach is based on statistical machine learning models. While these models can be induced computationally, they perform poorly because the complexity of the task far surpasses the amount of training data available.

We posit that statutory reasoning as presented to statistical models is underspecified, in that it was cast as Recognizing Textual Entailment (Dagan et al., 2005) and linear regression. Taking inspiration from the structure of Prolog programs, we re-frame statutory reasoning as a sequence of four tasks, prompting us to introduce a novel extension of the SARA dataset (Section 2), referred to as *SARA v2*. Beyond improving the model's performance, as shown in Section 3, the additional structure makes it more interpretable, and so more suitable for practical applications. We put our results in perspective in Section 4 and review related work in Section 5.

2 SARA v2

The symbolic solver requires experts translating the statutes and each new case's description into Prolog. In contrast, a machine learning-based model has the potential to generalize to unseen cases and to changing legislation, a significant advantage for a practical application. In the following, we argue that legal statutes share features with the symbolic solver's first-order logic. We formalize this connection in a series of four challenge tasks, described in this section, and depicted in Figure 1. We hope

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 2742–2758
August 1–6, 2021. ©2021 Association for Computational Linguistics

they provide structure to the problem, and a more efficient inductive bias for machine learning algorithms. The annotations mentioned throughout the remainder of this section were developed by the authors, entirely by hand, with regular guidance from a legal scholar[1]. Examples for each task are given in Appendix A. Statistics are shown in Figure 2 and further detailed in Appendix B.

Argument identification This first task, in conjunction with the second, aims to identify the arguments of the predicate that a given subsection represents. Some terms in a subsection refer to something concrete, such as "the United States" or "April 24th, 2017". Other terms can take a range of values depending on the case at hand, and act as placeholders. For example, in the top left box of Figure 1, the terms "a taxpayer" and "the taxable year" can take different values based on the context, while the terms "section 152" and "this paragraph" have concrete, immutable values. Formally, given a sequence of tokens $t_1, ..., t_n$, the task is to return a set of start and end indices $(s, e) \in \{1, 2, ..., n\}^2$ where each pair represents a span. We borrow from the terminology of predicate argument alignment (Roth and Frank, 2012; Wolfe et al., 2013) and call these placeholders *arguments*. The first task, which we call *argument identification*, is tagging which parts of a subsection denote such placeholders. We provide annotations for argument identification as character-level spans representing arguments. Since each span is a pointer to the corresponding argument, we made each span the shortest meaningful phrase. Figure 2(b) shows corpus statistics about placeholders.

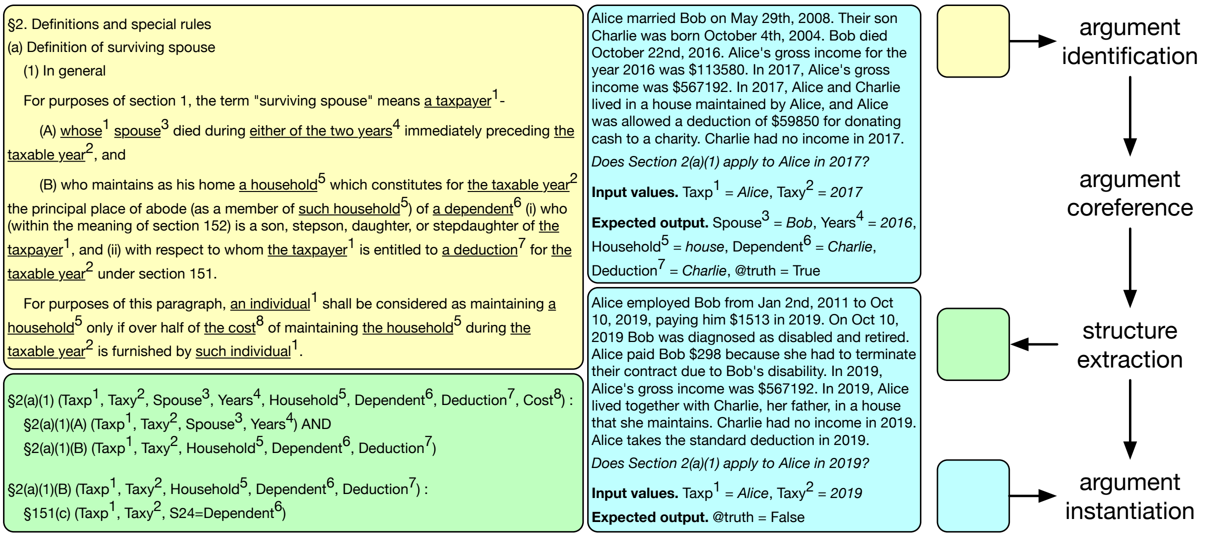
Argument coreference Some arguments detected in the previous task may appear multiple times within the same subsection. For instance, in the top left of Figure 1, the variable representing the taxpayer in §2(a)(1)(B) is referred to twice. We refer to the task of resolving this coreference problem at the level of the subsection as *argument coreference*. While this coreference can span across subsections, as is the case in Figure 1, we intentionally leave it to the next task. Keeping the notation of the above paragraph, given a set of spans $\{(s_i, e_i)\}_{i=1}^{S}$, the task is to return a matrix $C \in \{0, 1\}^{S \times S}$ where $C_{i,j} = 1$ if spans (s_i, e_i) and (s_j, e_j) denote the same variable, 0 otherwise. Corpus statistics about argument coreference can be found in Figure 2(a). After these first two tasks, we can extract a set of arguments for every subsection. In Figure 1, for §2(a)(1)(A), that would be {`Taxp`, `Taxy`, `Spouse`, `Years`}, as shown in the bottom left of Figure 1.

Structure extraction A prominent feature of legal statutes is the presence of references, implicit and explicit, to other parts of the statutes. Resolving references and their logical connections, and passing arguments appropriately from one subsection to the other, are major steps in statutory reasoning. We refer to this as *structure extraction*. This mapping can be trivial, with the taxpayer and taxable year generally staying the same across subsections. Some mappings are more involved, such as the taxpayer from §152(b)(1) becoming the dependent in §152(a). Providing annotations for this task in general requires expert knowledge, as many references are implicit, and some must be resolved using guidance from Treasury Regulations. Our approach contrasts with recent efforts in breaking down complex questions into atomic questions, with the possibility of referring to previous answers (Wolfson et al., 2020). Statutes contain their own breakdown into atomic questions. In addition, our structure is interpretable by a Prolog engine.

We provide structure extraction annotations for SARA in the style of Horn clauses (Horn, 1951), using common logical operators, as shown in the bottom left of Figure 1. We also provide character offsets for the start and end of each subsection. Argument identification and coreference, and structure extraction can be done with the statutes only. They correspond to extracting a shallow version of the symbolic solver of Holzenberger et al. (2020).

Argument instantiation We frame legal statutes as a set of predicates specified in natural language. Each subsection has a number of arguments, provided by the preceding tasks. Given the description of a case, each argument may or may not be associated with a value. Each subsection has an `@truth` argument, with possible values *True* or *False*, reflecting whether the subsection applies or not. Concretely, the input is (1) the string representation of the subsection, (2) the annotations from the first three tasks, and (3) values for some or all of its arguments. Arguments and values are represented as an array of key-value pairs, where the names of arguments specified in the structure an-

[1] The dataset can be found under `https://nlp.jhu.edu/law/`


§2. Definitions and special rules

(a) Definition of surviving spouse

(1) In general

For purposes of section 1, the term "surviving spouse" means a taxpayer^1 -

(A) whose^1 spouse^3 died during either of the two years^4 immediately preceding the taxable year^2, and

(B) who maintains as his home a household^5 which constitutes for the taxable year^2 the principal place of abode (as a member of such household^5) of a dependent^6 (i) who (within the meaning of section 152) is a son, stepson, daughter, or stepdaughter of the taxpayer^1, and (ii) with respect to whom the taxpayer^1 is entitled to a deduction^7 for the taxable year^2 under section 151.

For purposes of this paragraph, an individual^1 shall be considered as maintaining a household^5 only if over half of the cost^8 of maintaining the household^5 during the taxable year^2 is furnished by such individual^1.

§2(a)(1) (Taxp^1, Taxy^2, Spouse^3, Years^4, Household^5, Dependent^6, Deduction^7, Cost^8) :
§2(a)(1)(A) (Taxp^1, Taxy^2, Spouse^3, Years^4) AND
§2(a)(1)(B) (Taxp^1, Taxy^2, Household^5, Dependent^6, Deduction^7)

§2(a)(1)(B) (Taxp^1, Taxy^2, Household^5, Dependent^6, Deduction^7) :
§151(c) (Taxp^1, Taxy^2, S24=Dependent^6)

Alice married Bob on May 29th, 2008. Their son Charlie was born October 4th, 2004. Bob died October 22nd, 2016. Alice's gross income for the year 2016 was $113580. In 2017, Alice's gross income was $567192. In 2017, Alice and Charlie lived in a house maintained by Alice, and Alice was allowed a deduction of $59850 for donating cash to a charity. Charlie had no income in 2017.

Does Section 2(a)(1) apply to Alice in 2017?

Input values. Taxp^1 = *Alice*, Taxy^2 = *2017*

Expected output. Spouse^3 = *Bob*, Years^4 = *2016*, Household^5 = *house*, Dependent^6 = *Charlie*, Deduction^7 = *Charlie*, @truth = True

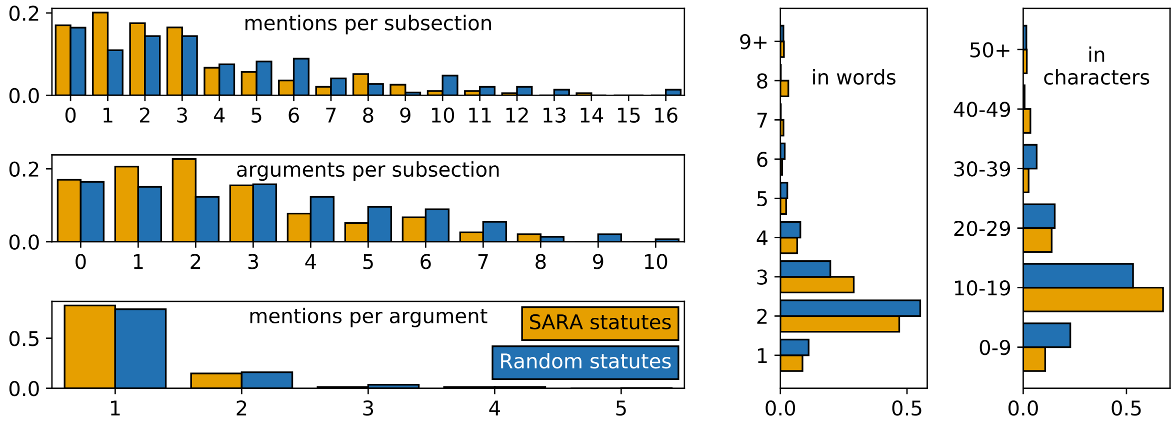
Alice employed Bob from Jan 2nd, 2011 to Oct 10, 2019, paying him $1513 in 2019. On Oct 10, 2019 Bob was diagnosed as disabled and retired. Alice paid Bob $298 because she had to terminate their contract due to Bob's disability. In 2019, Alice's gross income was $567192. In 2019, Alice lived together with Charlie, her father, in a house that she maintains. Charlie had no income in 2019. Alice takes the standard deduction in 2019.

Does Section 2(a)(1) apply to Alice in 2019?

Input values. Taxp^1 = *Alice*, Taxy^2 = *2019*

Expected output. @truth = False

Figure 1: Decomposing statutory reasoning into four tasks. The flowchart on the right indicates the ordering, inputs and outputs of the tasks. In the statutes in the yellow box, argument placeholders are underlined, and superscripts indicate argument coreference. The green box shows the logical structure of the statutes just above it. In blue are two examples of argument instantiation.



(a) Argument coreference

(b) Lengths of argument placeholders

Figure 2: Corpus statistics about arguments. “Random statutes” are 9 sections sampled from the US code.

notations are used as keys. In Figure 1, compare the names of arguments in the green box with the key names in the blue boxes. The output is values for its arguments, in particular for the `@truth` argument. In the example of the top right in Figure 1, the input values are `taxpayer` = *Alice* and `taxable year` = *2017*, and one expected output is `@truth` = True. We refer to this task as *argument instantiation*. Values for arguments can be found as spans in the case description, or must be predicted based on the case description. The latter happens often for dollar amounts, where incomes must be added, or tax must be computed. Figure 1 shows two examples of this task, in blue.

Before determining whether a subsection applies, it may be necessary to infer the values of unspecified arguments. For example, in the top of Figure 1, it is necessary to determine who Alice's deceased spouse and who the dependent mentioned in §2(a)(1)(B) are. If applicable, we provide values for these arguments, not as inputs, but as additional supervision for the model. We provide manual annotations for all (subsection, case) pairs in SARA. In addition, we run the Prolog solver of Holzenberger et al. (2020) to generate annotations for all possible (subsection, case) pairs, to be used as a *silver* standard, in contrast to the *gold* manual annotations. We exclude from the silver data any (subsection, case) pair where the case is part of the test set. This increases the amount of available training data by a factor of 210.

3 Baseline models

We provide baselines for three tasks, omitting structure extraction because it is the one task with the highest return on human annotation effort[2]. In other words, if humans could annotate for any of these four tasks, structure extraction is where we posit their involvement would be the most worthwhile. Further, Pertierra et al. (2017) have shown that the related task of semantic parsing of legal statutes is a difficult task, calling for a complex model.

3.1 Argument identification

We run the Stanford parser (Socher et al., 2013) on the statutes, and extract all noun phrases as spans – specifically, all NNP, NNPS, PRP$, NP and NML constituents. While de-formatting legal text can boost parser performance (Morgenstern, 2014), we found it made little difference in our case.

As an orthogonal approach, we train a BERT-based CRF model for the task of BIO tagging. With the 9 sections in the SARA v2 statutes, we create 7 equally-sized splits by grouping §68, 3301 and 7703 into a single split. We run a 7-fold cross-validation, using 1 split as a dev set, 1 split as a test set, and the remaining as training data. We embed each paragraph using BERT, classify each contextual subword embedding into a 3-dimensional logit with a linear layer, and run a CRF (Lafferty et al., 2001). The model is trained with gradient descent to maximize the log-likelihood of the sequence of gold tags. We experiment with using Legal BERT (Holzenberger et al., 2020) and BERT-base-cased (Devlin et al., 2019) as our BERT model. We freeze its parameters and optionally unfreeze the last layer. We use a batch size of 32 paragraphs, a learning rate of 10^{-3} and the Adam optimizer (Kingma and Ba, 2015). Based on F1 score measured on the dev set, the best model uses Legal BERT and unfreezes its last layer. Test results are shown in Table 1.

Parser-based	avg ± stddev	macro
precision	17.6 ± 4.4	16.6
recall	77.9 ± 5.0	77.3
F1	28.6 ± 6.2	27.3
BERT-based	avg ± stddev	macro
precision	64.7 ± 15.0	65.1
recall	69.0 ± 24.2	59.8
F1	66.2 ± 20.5	62.4

Table 1: Argument identification results. Average and standard deviations are computed across test splits.

3.2 Argument coreference

Argument coreference differs from the usual coreference task (Pradhan et al., 2014), even though we are using similar terminology, and frame it in a similar way. In argument coreference, it is equally as important to link two coreferent argument mentions as it is not to link two different arguments. In contrast, regular coreference emphasizes the prediction of links between mentions. We thus report a different metric in Tables 2 and 4, *exact match coreference*, which gives credit for returning a cluster of mentions that corresponds exactly to an argument. In Figure 1, a system would be rewarded for linking together both mentions of the taxpayer in §2(a)(1)(B), but not if any of the two mentions were linked to any other mention within §2(a)(1)(B). This custom metric gives as much credit for correctly linking a single-mention argument (no links), as for a 5-mention argument (10 links).

Single mention baseline Here, we predict no coreference links. Under usual coreference metrics, this system can have low performance.

String matching baseline This baseline predicts a coreference link if the placeholder strings of two arguments are identical, up to the presence of the words *such*, *a*, *an*, *the*, *any*, *his* and *every*.

Single mention	avg ± stddev	macro
precision	81.7 ± 28.9	68.2
recall	86.9 ± 21.8	82.7
F1	83.8 ± 26.0	74.8
String matching	avg ± stddev	macro
precision	91.2 ± 20.0	85.5
recall	92.8 ± 16.8	89.4
F1	91.8 ± 18.6	87.4

Table 2: Exact match coreference results. Average and standard deviations are computed across subsections.

We also provide usual coreference metrics in Table 3, using the code associated with Pradhan et al. (2014). This baseline perfectly resolves coreference for 80.8% of subsections, *versus* 68.9% for the single mention baseline.

[2] Code for the experiments can be found under https://github.com/SgfdDttt/sara_v2

	Single mention	String matching
MUC	0 / 0 / 0	82.1 / 64.0 / 71.9
CEAF$_m$	82.5 / 82.5 / 82.5	92.1 / 92.1 / 92.1
CEAF$_e$	77.3 / 93.7 / 84.7	90.9 / 95.2 / 93.0
BLANC	50.0 / 50.0 / 50.0	89.3 / 81.0 / 84.7

Table 3: Argument coreference baselines scored with usual metrics. Results are shown as Precision / Recall / F1.

In addition, we provide a cascade of the best methods for argument identification and coreference, and report results in Table 4. The cascade perfectly resolves a subsection's arguments in only 16.4% of cases. This setting, which groups the first two tasks together, offers a significant challenge.

Cascade	avg ± stddev	macro
precision	54.5 ± 35.6	58.0
recall	53.5 ± 37.2	52.4
F1	54.7 ± 33.4	55.1

Table 4: Exact match coreference results for BERT-based argument identification followed by string matching-based argument coreference. Average and standard deviations are computed across subsections.

3.3 Argument instantiation

Argument instantiation takes into account the information provided by previous tasks. We start by instantiating the arguments of a single subsection, without regard to the structure of the statutes. We then describe how the structure information is incorporated into the model.

Algorithm 1 Argument instantiation for a single subsection

```
Require: argument spans with coreference information $A$,
    input argument-value pairs $D$, subsection text $s$, case
    description $c$
Ensure: output argument-value pairs $P$
 1: function ArgInstantiation($A, D, s, c$)
 2:     $P \leftarrow \emptyset$
 3:     for $a$ in $A \setminus \{$@truth$\}$ do
 4:         $r \leftarrow$ InsertValues($s, A, D, P$)
 5:         $y \leftarrow$ BERT($c, r$)
 6:         $x \leftarrow$ ComputeAttentiveReps($y, a$)
 7:         $v \leftarrow$ PredictValue($x$)
 8:         $P \leftarrow P \cup (a, v)$
 9:     end for
10:     $r \leftarrow$ InsertValues($s, A, D, P$)
11:     $y \leftarrow$ BERT_CLS($c, r$)
12:     $t \leftarrow$ TruthPredictor($y$)
13:     $P \leftarrow P \cup ($@truth$, t)$
14: return $P$
15: end function
```

Single subsection We follow the paradigm of Chen et al. (2020), where we iteratively modify the text of the subsection by inserting argument values, and predict values for uninstantiated arguments. Throughout the following, we refer to Algorithm 1 and to its notation.

For each argument whose value is provided, we replace the argument's placeholders in subsection s by the argument's value, using InsertValues (line 4). This yields mostly grammatical sentences, with occasional hiccups. With §2(a)(1)(A) and the top right case from Figure 1, we obtain "(A) Alice spouse died during either of the two years immediately preceding 2017".

We concatenate the text of the case c with the modified text of the subsection r, and embed it using BERT (line 5), yielding a sequence of contextual subword embeddings $y = \{y_i \in \mathbb{R}^{768} \,|\, i = 1...n\}$. Keeping with the notation of Chen et al. (2020), assume that the embedded case is represented by the sequence of vectors $\boldsymbol{t}_1, ..., \boldsymbol{t}_m$ and the embedded subsection by $\boldsymbol{s}_1, ..., \boldsymbol{s}_n$. For a given argument a, compute its attentive representation $\tilde{\boldsymbol{s}}_1, ..., \tilde{\boldsymbol{s}}_m$ and its augmented feature vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_m$. This operation, described by Chen et al. (2020), is performed by ComputeAttentiveReps (line 6). The augmented feature vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_m$ represent the argument's placeholder, conditioned on the text of the statute and case.

Based on the name of the argument span, we predict its value v either as an integer or a span from the case description, using PredictValue (line 7). For integers, as part of the model training, we run k-means clustering on the set of all integer values in the training set, with enough centroids such that returning the closest centroid instead of the true value yields a numerical accuracy of 1 (see below). For any argument requiring an integer (e.g. `tax`), the model returns a weighted average of the centroids. The weights are predicted by a linear layer followed by a softmax, taking as input an average-pooling and a maxpooling of $\boldsymbol{x}_1, ..., \boldsymbol{x}_m$. For a span from the case description, we follow the standard procedure for fine-tuning BERT on SQuAD (Devlin et al., 2019). The unnormalized probability of the span from tokens i to j is given by $e^{\boldsymbol{l} \cdot \boldsymbol{x}_i + \boldsymbol{r} \cdot \boldsymbol{x}_j}$ where $\boldsymbol{l}, \boldsymbol{r}$ are learnable parameters.

The predicted value v is added to the set of predictions P (line 8), and will be used in subsequent iterations to replace the argument's placeholder in the subsection. We repeat this process until a value has been predicted for every argument, except @truth (lines 3-9). Arguments are processed in order of appearance in the subsection. Finally, we concatenate the case and fully grounded subsection and embed them with BERT (lines 10-11), then use a linear predictor on top of the representation for the [CLS] token to predict the value for the @truth argument (line 12).

Algorithm 2 Argument instantiation with dependencies

```
Require: argument spans with coreference information $A$,
    structure information $T$, input argument-value pairs $D$,
    subsection $s$, case description $c$
Ensure: output argument-value pairs $P$
 1: function ArgInstantiationFull($A, T, D, s, c$)
 2:     $t \leftarrow$ BuildDependencyTree($s, T$)
 3:     $t \leftarrow$ PopulateArgValues($t, D$)
 4:     $Q \leftarrow$ depth-first traversal of $t$
 5:     for $q$ in $Q$ do
 6:         if $q$ is a subsection and a leaf node then
 7:             $D_q \leftarrow$ GetArgValuePairs($q$)
 8:             $\tilde{s} \leftarrow$ GetSubsectionText($q$)
 9:             $q \leftarrow$ ArgInstantiation($A, D_q, \tilde{s}, c$)
10:         else if $q$ is a subsection and not a leaf node then
11:             $D_q \leftarrow$ GetArgValuePairs($q$)
12:             $x \leftarrow$ GetChild($q$)
13:             $D_x \leftarrow$ GetArgValuePairs($x$)
14:             $D_q \leftarrow D_{\mathrm{q}} \cup D_x$
15:             $\tilde{s} \leftarrow$ GetSubsectionText($q$)
16:             $q \leftarrow$ ArgInstantiation($A, D_q, \tilde{s}, c$)
17:         else if $q \in \{\mathrm{AND}, \mathrm{OR}, \mathrm{NOT}\}$ then
18:             $C \leftarrow$ GetChildren($q$)
19:             $q \leftarrow$ DoOperation($C, q$)
20:         end if
21:     end for
22:     $x \leftarrow$ Root($t$)
23:     $P \leftarrow$ GetArgValuePairs($x$)
24: return $P$
25: end function
```

Subsection with dependencies To describe our procedure at a high-level, we use the structure of the statutes to build out a computational graph, where nodes are either subsections with argument-value pairs, or logical operations. We resolve nodes one by one, depth first. We treat the single-subsection model described above as a function, taking as input a set of argument-value pairs, a string representation of a subsection, and a string representation of a case, and returning a set of argument-value pairs. Algorithm 2 and Figure 3 summarize the following.

We start by building out the subsection's dependency tree, as specified by the structure annotations (lines 2-4). First, we build the tree structure using BuildDependencyTree. Then, values for arguments are propagated from parent to child, from the root down, with PopulateArgValues. The tree is optionally capped to a predefined depth. Each node is either an input for the single-subsection function or its output, or a logical operation. We then traverse the tree depth first, performing the following operations, and replacing the node with the result of the operation:

- If the node q is a leaf, resolve it using the single-subsection function ArgInstantiation (lines 6-9 in Algorithm 2; step 1 in Figure 3).

- If the node q is a subsection that is not a leaf, find its child node x (GetChild, line 12), and corresponding argument-value pairs other than @truth, D_x (GetArgValuePairs, line 13). Merge D_x with D_q, the argument-value pairs of the main node q (line 14). Finally, resolve the parent node q using the single-subsection function (lines 15-16; step 3 in Figure 3.

- If node q is a logical operation (line 17), get its children C (GetChildren, line 18), to which the operation will be applied with DoOperation (line 19) as follows:
  - If $q ==$ NOT, assign the negation of the child's @truth value to q.
  - If $q ==$ OR, pick its child with the highest @truth value, and assign its arguments' values to q.
  - If $q ==$ AND, transfer the argument-value pairs from all its children to q. In case of conflicting values, use the value associated with the lower @truth value. This operation can be seen in step 4 of Figure 3.

This procedure follows the formalism of neural module networks (Andreas et al., 2016) and is illustrated in Figure 3. Reentrancy into the dependency tree is not possible, so that a decision made earlier cannot be backtracked on at a later stage. One could imagine doing joint inference, or using heuristics for revisiting decisions, for example with a limited number of reentrancies. Humans are generally able to resolve this task in the order of the text, and we assume it should be possible for a computational model too. Our solution is meant to be computationally efficient, with the hope of not sacrificing too much performance. Revisiting this assumption is left for future work.

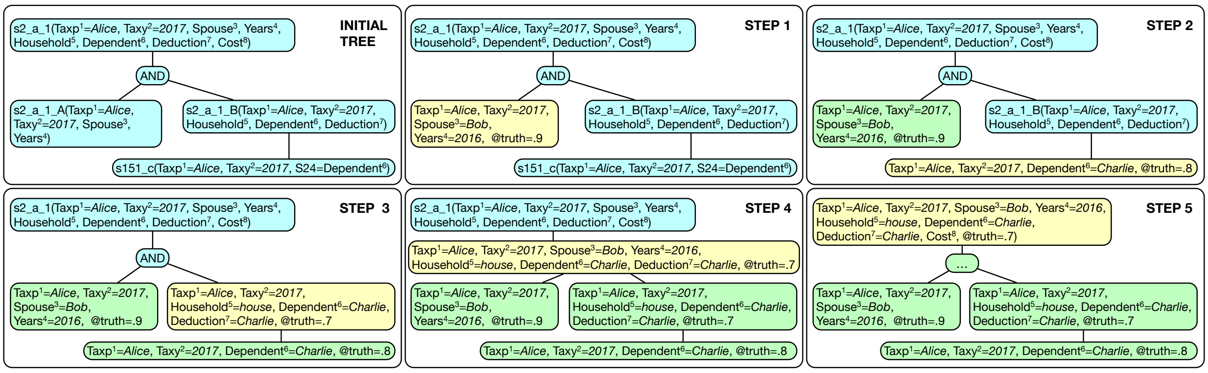

INITIAL TREE
s2_a_1(Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3, Years^4, Household^5, Dependent^6, Deduction^7, Cost^8)
AND
s2_a_1_A(Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3, Years^4)
s2_a_1_B(Taxp^1=*Alice*, Taxy^2=*2017*, Household^5, Dependent^6, Deduction^7)
s151_c(Taxp^1=*Alice*, Taxy^2=*2017*, S24=Dependent^6)
STEP 1
s2_a_1(Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3, Years^4, Household^5, Dependent^6, Deduction^7, Cost^8)
AND
Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3=*Bob*, Years^4=*2016*, @truth=.9
s2_a_1_B(Taxp^1=*Alice*, Taxy^2=*2017*, Household^5, Dependent^6, Deduction^7)
s151_c(Taxp^1=*Alice*, Taxy^2=*2017*, S24=Dependent^6)
STEP 2
s2_a_1(Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3, Years^4, Household^5, Dependent^6, Deduction^7, Cost^8)
AND
Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3=*Bob*, Years^4=*2016*, @truth=.9
s2_a_1_B(Taxp^1=*Alice*, Taxy^2=*2017*, Household^5, Dependent^6, Deduction^7)
Taxp^1=*Alice*, Taxy^2=*2017*, Dependent^6=*Charlie*, @truth=.8
STEP 3
s2_a_1(Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3, Years^4, Household^5, Dependent^6, Deduction^7, Cost^8)
AND
Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3=*Bob*, Years^4=*2016*, @truth=.9
Taxp^1=*Alice*, Taxy^2=*2017*, Household^5=*house*, Dependent^6=*Charlie*, Deduction^7=*Charlie*, @truth=.7
Taxp^1=*Alice*, Taxy^2=*2017*, Dependent^6=*Charlie*, @truth=.8
STEP 4
s2_a_1(Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3, Years^4, Household^5, Dependent^6, Deduction^7, Cost^8)
Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3=*Bob*, Years^4=*2016*, Household^5=*house*, Dependent^6=*Charlie*, Deduction^7=*Charlie*, @truth=.7
Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3=*Bob*, Years^4=*2016*, @truth=.9
Taxp^1=*Alice*, Taxy^2=*2017*, Household^5=*house*, Dependent^6=*Charlie*, Deduction^7=*Charlie*, @truth=.7
Taxp^1=*Alice*, Taxy^2=*2017*, Dependent^6=*Charlie*, @truth=.8
STEP 5
Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3=*Bob*, Years^4=*2016*, Household^5=*house*, Dependent^6=*Charlie*, Deduction^7=*Charlie*, Cost^8, @truth=.7)
...
Taxp^1=*Alice*, Taxy^2=*2017*, Spouse^3=*Bob*, Years^4=*2016*, @truth=.9
Taxp^1=*Alice*, Taxy^2=*2017*, Household^5=*house*, Dependent^6=*Charlie*, Deduction^7=*Charlie*, @truth=.7
Taxp^1=*Alice*, Taxy^2=*2017*, Dependent^6=*Charlie*, @truth=.8

Figure 3: Argument instantiation with the top example from Figure 1. At each step, nodes to be processed are in blue, nodes being processed in yellow, and nodes already processed in green. The last step was omitted, and involves determining the truth value of the root node's `@truth` argument.

Metrics and evaluation Arguments whose value needs to be predicted fall into three categories. The `@truth` argument calls for a binary truth value, and we score a model's output using binary accuracy. The values of some arguments, such as `gross income`, are dollar amounts. We score such values using numerical accuracy, as 1 if $\Delta(y, \hat{y}) = \frac{|y-\hat{y}|}{\max(0.1*y, 5000)} < 1$ else 0, where $\hat{y}$ is the prediction and y the target. All other argument values are treated as strings. In those cases, we compute accuracy as exact match between predicted and gold value. Each of these three metrics defines a form of accuracy. We average the three metrics, weighted by the number of samples, to obtain a unified accuracy metric, used to compare the performance of models.

Training Based on the type of value expected, we use different loss functions. For `@truth`, we use binary cross-entropy. For numerical values, we use the hinge loss $\max(\Delta(y, \hat{y}) - 1, 0)$. For strings, let S be all the spans in the case description equal to the expected value. The loss function is $\log(\sum_{i \leq j} e^{\boldsymbol{l} \cdot \boldsymbol{x}_i + \boldsymbol{r} \cdot \boldsymbol{x}_j}) - \log(\sum_{i,j \in S} e^{\boldsymbol{l} \cdot \boldsymbol{x}_i + \boldsymbol{r} \cdot \boldsymbol{x}_j})$ (Clark and Gardner, 2018). The model is trained end-to-end with gradient descent.

We start by training models on the silver data, as a pre-training step. We sweep the values of the learning rate in $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ and the batch size in $\{64, 128, 256\}$. We try both BERT-base-cased and Legal BERT, allowing updates to the parameters of its top layer. We set aside 10% of the silver data as a dev set, and select the best model based on the unified accuracy on the dev set. Training is split up into three stages. The single-subsection model iteratively inserts values for arguments into the text of the subsection. In the first stage, regardless of the predicted value, we insert the gold value for the argument, as in teacher forcing (Kolen and Kremer, 2001). In the second and third stages, we insert the value predicted by the model. When initializing the model from one stage to the next, we pick the model with the highest unified accuracy on the dev set. In the first two stages, we ignore the structure of the statutes, which effectively caps the depth of each dependency tree at 1.

Picking the best model from this pre-training step, we perform fine-tuning on the gold data. We take a k-fold cross-validation approach (Stone, 1974). We randomly split the SARA v2 training set into 10 splits, taking care to put pairs of cases testing the same subsection into the same split. Each split contains nearly exactly the same proportion of binary and numerical cases. We sweep the values of the learning rate and batch size in the same ranges as above, and optionally allow updates to the parameters of BERT's top layer. For a given set of hyperparameters, we run training on each split, using the dev set and the unified metric for early stopping. We use the performance on the dev set averaged across the 10 splits to evaluate the performance of a given set of hyperparameters. Using that criterion, we pick the best set of hyperparameters. We then pick the final model as that which achieves median performance on the dev set, across the 10 splits. We report the performance of that model on the test set.

In Table 5, we report the relevant argument instantiation metrics, under `@truth`, `dollar amount` and `string`. For comparison, we also report binary and numerical accuracy metrics defined in Holzenberger et al. (2020). The reported

	@truth	dollar amount	string	unified	binary	numerical
baseline	58.3 ± 7.5	18.2 ± 11.5	4.4 ± 7.4	43.3 ± 6.2	50 ± 8.3	30 ± 18.1
+ silver	58.3 ± 7.5	**39.4** ± 14.6	4.4 ± 7.4	47.2 ± 6.2	50 ± 8.3	**45** ± 19.7
BERT	59.2 ± 7.5	23.5 ± 12.5	**37.5** ± 17.3	49.4 ± 6.2	51 ± 8.3	30 ± 18.1
- pre-training	57.5 ± 7.5	20.6 ± 11.9	**37.5** ± 17.3	47.8 ± 6.2	49 ± 8.3	30 ± 18.1
- structure	**65.8** ± 7.2	20.6 ± 11.9	33.3 ± 16.8	**52.8** ± 6.2	**59** ± 8.2	30 ± 18.1
- pre-training, structure	60.8 ± 7.4	20.6 ± 11.9	33.3 ± 16.8	49.4 ± 6.2	53 ± 8.3	30 ± 18.1

(best results in **bold**)

Table 5: Argument instantiation. We report accuracies, in %, and the 90% confidence interval. Right of the bar are accuracy metrics proposed with the initial release of the dataset. Blue cells use the silver data, brown cells do not. "BERT" is the model described in Section 3.3. Ablations to it are marked with a "-" sign.

baseline has three parameters. For @truth, it returns the most common value for that argument on the train set. For arguments that call for a dollar amount, it returns the one number that minimizes the dollar amount hinge loss on the training set. For all other arguments, it returns the most common string answer in the training set. Those parameters vary depending on whether the training set is augmented with the silver data.

4 Discussion

Our goal in providing the baselines of Section 3 is to identify performance bottlenecks in the proposed sequence of tasks. Argument identification poses a moderate challenge, with a language model-based approach achieving non-trivial F1 score. The simple parser-based method is not a sufficient solution, but with its high recall could serve as the backbone to a statistical method. Argument coreference is a simpler task, with string matching perfectly resolving nearly 80% of the subsections. This is in line with the intuition that legal language is very explicit about disambiguating coreference. As reported in Table 3, usual coreference metrics seem lower, but only reflect a subset of the full task: coreference metrics are only concerned with links, so that arguments appearing exactly once bear no weight under that metric, unless they are wrongly linked to another argument.

Argument instantiation is by far the most challenging task, as the model needs strong natural language understanding capabilities. Simple baselines can achieve accuracies above 50% for @truth, since for all numerical cases, @truth = True. We receive a slight boost in binary accuracy from using the proposed paradigm, departing from previous results on this benchmark. As compared to the baseline, the models mostly lag behind for the dollar amount and numerical accuracies, which can be explained by the lack of a dedicated numerical solver, and sparse data. Further, we have made a number of simplifying assumptions, which may be keeping the model from taking advantage of the structure information: arguments are instantiated in order of appearance, forbidding joint prediction; revisiting past predictions is disallowed, forcing the model to commit to wrong decisions made earlier; the depth of the dependency tree is capped at 3; and finally, information is being passed along the dependency tree in the form of argument values, as opposed to dense, high-dimensional vector representations. The latter limits both the flow of information and the learning signal. This could also explain why the use of dependencies is detrimental in some cases. Future work would involve joint prediction (Chan et al., 2019), and more careful use of structure information.

Looking at the errors made by the best model in Table 5 for binary accuracy, we note that for 39 positive and negative case pairs, it answers each pair identically, thus yielding 39 correct answers. In the remaining 11 pairs, there are 10 pairs where it gets both cases right. This suggests it may be guessing randomly on 39 pairs, and understanding 10. The best BERT-based model for dollar amounts predicts the same number for each case, as does the baseline. The best models for string arguments generally make predictions that match the category of the expected answer (date, person, etc) while failing to predict the correct string.

Performance gains from silver data are noticeable and generally consistent, as can be seen by comparing brown and blue cells in Table 5. The silver data came from running a human-written Prolog program, which is costly to produce. A possible substitute is to find mentions of applicable statutes in large corpora of legal cases (Caselaw, 2019), for example using high-precision rules (Ratner et al., 2017), which has been successful for extracting information from cases (Boniol et al., 2020).

In this work, each task uses the gold annotations from upstream tasks. Ultimately, the goal is to pass the outputs of models from one task to the next.

5 Related Work

Law-related NLP tasks have flourished in the past years, with applications including answering bar exam questions (Yoshioka et al., 2018; Zhong et al., 2020), information extraction (Chalkidis et al., 2019b; Boniol et al., 2020; Lam et al., 2020), managing contracts (Elwany et al., 2019; Liepiņa et al., 2020; Nyarko, 2021) and analyzing court decisions (Sim et al., 2015; Lee and Mouritsen, 2017). Case-based reasoning has been approached with expert systems (Popp and Schlink, 1974; Hellawell, 1980; v. d. L. Gardner, 1983), high-level hand-annotated features (Ashley and Brüninghaus, 2009) and transformer-based models (Rabelo et al., 2019). Closest to our work is Saeidi et al. (2018), where a dialog agent's task is to answer a user's question about a set of regulations. The task relies on a set of questions provided within the dataset.

Clark et al. (2019) as well as preceding work (Friedland et al., 2004; Gunning et al., 2010) tackle a similar problem in the science domain, with the goal of using the prescriptive knowledge from science textbooks to answer exam questions. The core of their model relies on several NLP and specialized reasoning techniques, with contextualized language models playing a major role. Clark et al. (2019) take the route of sorting questions into different types, and working on specialized solvers. In contrast, our approach is to treat each question identically, but to decompose the process of answering into a sequence of subtasks.

The language of statutes is related to procedural language, which describes steps in a process. Zhang et al. (2012) collect how-to instructions in a variety of domains, while Wambsganss and Fromm (2019) focus on automotive repair instructions. Branavan et al. (2012) exploit instructions in a game manual to improve an agent's performance. Dalvi et al. (2019) and Amini et al. (2020) turn to modeling textual descriptions of physical and biological mechanisms. Weller et al. (2020) propose models that generalize to new task descriptions.

The tasks proposed in this work are germane to standard NLP tasks, such as named entity recognition (Ratinov and Roth, 2009), part-of-speech tagging (Petrov et al., 2012; Akbik et al., 2018), and coreference resolution (Pradhan et al., 2014). Structure extraction is conceptually similar to syntactic (Socher et al., 2013) and semantic parsing (Berant et al., 2013), which Pertierra et al. (2017) attempt for a subsection of tax law.

Argument instantiation is closest to the task of aligning predicate argument structures (Roth and Frank, 2012; Wolfe et al., 2013). We frame argument instantiation as iteratively completing a statement in natural language. Chen et al. (2020) refine generic statements by copying strings from input text, with the goal of detecting events. Chan et al. (2019) extend transformer-based language models to permit inserting tokens anywhere in a sequence, thus allowing to modify an existing sequence. For argument instantiation, we make use of neural module networks (Andreas et al., 2016), which are used in the visual (Yi et al., 2018) and textual domains (Gupta et al., 2020). In that context, arguments and their values can be thought of as the hints from Khot et al. (2020). The Prolog-based data augmentation is related to data augmentation for semantic parsing (Campagna et al., 2019; Weir et al., 2019).

6 Conclusion

Solutions to tackle statutory reasoning may range from high-structure, high-human involvement expert systems, to less structured, largely self-supervised language models. Here, taking inspiration from Prolog programs, we introduce a novel paradigm, by breaking statutory reasoning down into a sequence of tasks. Each task can be annotated for with far less expertise than would be required to translate legal language into code, and comes with its own performance metrics. Our contribution enables finer-grained scoring and debugging of models for statutory reasoning, which facilitates incremental progress and identification of performance bottlenecks. In addition, argument instantiation and explicit resolution of dependencies introduce further interpretability. This novel approach could possibly inform the design of models that reason with rules specified in natural language, for the domain of legal NLP and beyond.

Acknowledgments

The authors thank Andrew Blair-Stanek for helpful comments, and Ryan Culkin for help with the parser-based argument identification baseline.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1638–1649. Association for Computational Linguistics.

Aida Amini, Antoine Bosselut, Bhavana Dalvi Mishra, Yejin Choi, and Hannaneh Hajishirzi. 2020. Procedural reading comprehension with attribute-aware context flow. In *Conference on Automated Knowledge Base Construction, AKBC 2020, Virtual, June 22-24, 2020*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 39–48. IEEE Computer Society.

Kevin D. Ashley and Stefanie Brüninghaus. 2009. Automatically classifying case texts and predicting outcomes. *Artif. Intell. Law*, 17(2):125–165.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Paul Boniol, George Panagopoulos, Christos Xypolopoulos, Rajaa El Hamdani, David Restrepo Amariles, and Michalis Vazirgiannis. 2020. Performance in the courtroom: Automated processing and visualization of appeal court decisions in france. In *Proceedings of the Natural Legal Language Processing Workshop 2020 co-located with the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2020), Virtual Workshop, August 24, 2020*, volume 2645 of *CEUR Workshop Proceedings*, pages 11–17. CEUR-WS.org.

S.R.K. Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning high-level planning from text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 126–135, Jeju Island, Korea. Association for Computational Linguistics.

Giovanni Campagna, Silei Xu, Mehrad Moradshahi, Richard Socher, and Monica S. Lam. 2019. Genie: a generator of natural language semantic parsers for virtual assistant commands. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019*, pages 394–410. ACM.

Caselaw. 2019. Caselaw access project.

Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. 2019a. Neural legal judgment prediction in English. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4317–4323, Florence, Italy. Association for Computational Linguistics.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2019b. Large-scale multi-label text classification on EU legislation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6314–6322. Association for Computational Linguistics.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019. KERMIT: generative insertion-based modeling for sequences. *CoRR*, abs/1906.01604.

Yunmo Chen, Tongfei Chen, Seth Ebner, Aaron Steven White, and Benjamin Van Durme. 2020. Reading the manual: Event extraction as definition comprehension. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP@EMNLP 2020, Online, November 20, 2020*, pages 74–83. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 845–855. Association for Computational Linguistics.

Peter Clark, Oren Etzioni, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, Sumithra Bhakthavatsalam, Dirk Groeneveld, Michal Guerquin, and Michael Schmitz. 2019. From 'f' to 'a' on the N.Y. regents science exams: An overview of the aristo project. *CoRR*, abs/1909.01958.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, and Peter Clark. 2019. Everything happens for a reason: Discovering the purpose of actions in procedural text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4496–4505, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Emad Elwany, Dave Moore, and Gaurav Oberoi. 2019. BERT goes to law school: Quantifying the competitive advantage of access to large legal corpora in contract understanding. *CoRR*, abs/1911.00473.

Noah S. Friedland, Paul G. Allen, Gavin Matthews, Michael J. Witbrock, David Baxter, Jon Curtis, Blake Shepard, Pierluigi Miraglia, Jürgen Angele, Steffen Staab, Eddie Mönch, Henrik Oppermann, Dirk Wenke, David J. Israel, Vinay K. Chaudhri, Bruce W. Porter, Ken Barker, James Fan, Shaw Yi Chaw, Peter Z. Yeh, Dan Tecuci, and Peter Clark. 2004. Project halo: Towards a digital aristotle. *AI Mag.*, 25(4):29–48.

David Gunning, Vinay K. Chaudhri, Peter Clark, Ken Barker, Shaw Yi Chaw, Mark Greaves, Benjamin N. Grosof, Alice Leung, David D. McDonald, Sunil Mishra, John Pacheco, Bruce W. Porter, Aaron Spaulding, Dan Tecuci, and Jing Tien. 2010. Project halo update - progress toward digital aristotle. *AI Mag.*, 31(3):33–58.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2020. Neural module networks for reasoning over text. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Robert Hellawell. 1980. A computer program for legal planning and analysis: Taxation of stock redemptions. *Columbia Law Review*, 80(7):1363–1398.

Nils Holzenberger, Andrew Blair-Stanek, and Benjamin Van Durme. 2020. A dataset for statutory reasoning in tax law entailment and question answering. In *Proceedings of the Natural Legal Language Processing Workshop 2020 co-located with the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2020), Virtual Workshop, August 24, 2020*, volume 2645 of *CEUR Workshop Proceedings*, pages 31–38. CEUR-WS.org.

Alfred Horn. 1951. On sentences which are true of direct unions of algebras. *J. Symb. Log.*, 16(1):14–21.

Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2020. Text modular networks: Learning to decompose tasks in the language of existing models. *CoRR*, abs/2009.00751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

John F Kolen and Stefan C Kremer. 2001. *A field guide to dynamical recurrent networks*. John Wiley & Sons.

Anne v. d. L. Gardner. 1983. The design of a legal analysis program. In *Proceedings of the National Conference on Artificial Intelligence, Washington, D.C., USA, August 22-26, 1983*, pages 114–118. AAAI Press.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289. Morgan Kaufmann.

Jason T. Lam, David Liang, Samuel Dahan, and Farhana H. Zulkernine. 2020. The gap between deep learning and law: Predicting employment notice. In *Proceedings of the Natural Legal Language Processing Workshop 2020 co-located with the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2020), Virtual Workshop, August 24, 2020*, volume 2645 of *CEUR Workshop Proceedings*, pages 52–56. CEUR-WS.org.

Thomas R Lee and Stephen C Mouritsen. 2017. Judging ordinary meaning. *Yale LJ*, 127:788.

Rūta Liepiņa, Federico Ruggeri, Francesca Lagioia, Marco Lippi, Kasper Drazewski, and Paolo Torroni. 2020. Explaining potentially unfair clauses to the consumer with the CLAUDETTE tool. In *Proceedings of the Natural Legal Language Processing Workshop 2020 co-located with the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2020), Virtual Workshop, August 24, 2020*, volume 2645 of *CEUR Workshop Proceedings*, pages 61–64. CEUR-WS.org.

Leora Morgenstern. 2014. Toward automated international law compliance monitoring (tailcm). Technical report, LEIDOS HOLDINGS INC RESTON VA.

Julian Nyarko. 2021. Stickiness and incomplete contracts. *The University of Chicago Law Review*, 88.

Marcos A. Pertierra, Sarah Lawsky, Erik Hemberg, and Una-May O'Reilly. 2017. Towards formalizing statute law as default logic through automatic semantic parsing. In *Proceedings of the Second Workshop on Automated Semantic Analysis of Information in Legal Texts co-located with the 16th International Conference on Artificial Intelligence and*

Law (ICAIL 2017), London, UK, June 16, 2017, volume 2143 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Slav Petrov, Dipanjan Das, and Ryan T. McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 2089–2096. European Language Resources Association (ELRA).

Walter G Popp and Bernhard Schlink. 1974. Judith, a computer program to advise lawyers in reasoning a case. *Jurimetrics J.*, 15:303.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–35, Baltimore, Maryland. Association for Computational Linguistics.

Juliano Rabelo, Mi-Young Kim, and Randy Goebel. 2019. Combining similarity and transformer methods for case law entailment. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law, ICAIL 2019, Montreal, QC, Canada, June 17-21, 2019*, pages 290–296. ACM.

Lev-Arie Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning, CoNLL 2009, Boulder, Colorado, USA, June 4-5, 2009*, pages 147–155. ACL.

Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proc. VLDB Endow.*, 11(3):269–282.

Michael Roth and Anette Frank. 2012. Aligning predicate argument structures in monolingual comparable texts: A new corpus for a new task. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics, *SEM 2012, June 7-8, 2012, Montréal, Canada*, pages 218–227. Association for Computational Linguistics.

Marzieh Saeidi, Max Bartolo, Patrick S. H. Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2087–2097. Association for Computational Linguistics.

Yanchuan Sim, Bryan R. Routledge, and Noah A. Smith. 2015. The utility of text: The case of amicus briefs and the supreme court. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2311–2317. AAAI Press.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 455–465. The Association for Computer Linguistics.

Mervyn Stone. 1974. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133.

Thomas Vacek, Ronald Teo, Dezhao Song, Timothy Nugent, Conner Cowling, and Frank Schilder. 2019. Litigation analytics: Case outcomes extracted from US federal court dockets. In *Proceedings of the Natural Legal Language Processing Workshop 2019*, pages 45–54, Minneapolis, Minnesota. Association for Computational Linguistics.

Thiemo Wambsganss and Hansjörg Fromm. 2019. Mining user-generated repair instructions from automotive web communities. In *52nd Hawaii International Conference on System Sciences, HICSS 2019, Grand Wailea, Maui, Hawaii, USA, January 8-11, 2019*, pages 1–10. ScholarSpace.

Nathaniel Weir, Andrew Crotty, Alex Galakatos, Amir Ilkhechi, Shekar Ramaswamy, Rohin Bhushan, Ugur Çetintemel, Prasetya Utama, Nadja Geisler, Benjamin Hättasch, Steffen Eger, and Carsten Binnig. 2019. Dbpal: Weak supervision for learning a natural language interface to databases. *CoRR*, abs/1909.06182.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E. Peters. 2020. Learning from task descriptions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1361–1375. Association for Computational Linguistics.

Travis Wolfe, Benjamin Van Durme, Mark Dredze, Nicholas Andrews, Charley Beller, Chris Callison-Burch, Jay DeYoung, Justin Snyder, Jonathan Weese, Tan Xu, and Xuchen Yao. 2013. PARMA: A predicate argument aligner. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 63–68, Sofia, Bulgaria. Association for Computational Linguistics.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

Chaojun Xiao, Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Zhiyuan Liu, Maosong Sun, Yansong Feng, Xianpei Han, Zhen Hu, Heng Wang, and Jianfeng Xu. 2018. CAIL2018: A large-scale legal dataset for judgment prediction. *CoRR*, abs/1807.02478.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-symbolic VQA: disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 1039–1050.

Masaharu Yoshioka, Yoshinobu Kano, Naoki Kiyota, and Ken Satoh. 2018. Overview of japanese statute law retrieval and entailment task at coliee-2018. In *Twelfth international workshop on Juris-informatics (JURISIN 2018)*.

Ziqi Zhang, Philip Webster, Victoria S. Uren, Andrea Varga, and Fabio Ciravegna. 2012. Automatically extracting procedural knowledge from instructional texts using natural language processing. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 520–527. European Language Resources Association (ELRA).

Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. 2020. JEC-QA: A legal-domain question answering dataset. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9701–9708. AAAI Press.

A Task examples

In the following, we provide several examples for each of the tasks defined in Section 2.

A.1 Argument identification

For ease of reading, the spans mentioned in the output are underlined in the input.

Input 1 (§3306(a)(1)(B))

(B) on each of some 10 days during the calendar year or during the preceding calendar year, each day being in a different calendar week, employed at least one individual in employment for some portion of the day.

Output 1

$\{(15, 26), (35, 51), (62, 88), (92, 99), (122, 134), (155, 168), (173, 182), (188, 210)\}$

Input 2 (§63(c)(5))

In the case of an individual with respect to whom a deduction under section 151 is allowable to another taxpayer for a taxable year beginning in the calendar year in which the individual's taxable year begins, the basic standard deduction applicable to such individual for such individual's taxable year shall not exceed the greater of-

Output 2

$\{(15, 27), (50, 60), (96, 111), (117, 130), (145, 161), (172, 185), (189, 200), (210, 237), (253, 267), (273, 287), (291, 302), (321, 331)\}$

Input 3 (§1(d)(iv))

(iv) $31,172, plus 36% of the excess over $115,000 if the taxable income is over $115,000 but not over $250,000;

Output 3

$\{(5, 45), (50, 67)\}$

A.2 Argument coreference

We report the full matrix C. In addition, for ease of reading, coreference clusters are marked with superscripts in the input.

Input 1 (§3306(a)(1)(B))

(B) on each of some 10 days^1 during the calendar year^2 or during the preceding calendar year^3, each day^1 being in a different calendar week^4, employed at least one individual^5 in employment^6 for some portion of the day^7.

$\{(15, 26), (35, 51), (62, 88), (92, 99), (122, 134), (155, 168), (173, 182), (188, 210)\}$

Output 1

$$\begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Input 2 (§63(c)(5))

In the case of an individual^1 with respect to whom a deduction^2 under section 151 is allowable to another taxpayer^3 for a taxable year^4 beginning in the calendar year^5 in which the individual^1's taxable year^6 begins, the basic standard deduction^7 applicable to such individual^1 for such individual^1's taxable year^6 shall not exceed the greater^8 of-
$\{(15, 27), (50, 60), (96, 111), (117, 130), (145, 161), (172, 185), (189, 200), (210, 237), (253, 267), (273, 287), (291, 302), (321, 331)\}$

Output 2

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Input 3 (§1(d)(iv))

(iv) $31,172, plus 36% of the excess over $115,000^1 if the taxable income^2 is over $115,000 but not over $250,000;
$\{(5, 45), (50, 67)\}$

Output 3

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

A.3 Structure extraction

To clarify the link between the input and the output, we are adding superscripts to argument names in the output. While the output is represented as plain text, a graph-based representation would likely be used in a practical system, to facilitate learning and inference. Arguments are keyword based. For example, in Output 2, the value of the Taxp argument of §63(c)(5) is passed to the Spouse argument of §151(b). If no equal sign is specified, it means the argument names match. For example, part of Output 2 could have been rewritten more explicitly as §151(b)(Spouse=Taxp, Taxp=S45, Taxy=Taxy).

Input 1 (§3306(a)(1)(B))

(B) on each of some 10 days^1 during the calendar year^2 or during the preceding calendar year^3, each day^1 being in a different calendar week^4, employed at least one individual^5 in employment^6 for some portion of the day^7.
$\{(15,26),(35,51),(62,88),(92,99),(122,134),(155,168),(173,182),(188,210)\}$

$$\begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Output 1

§3306(a)(1)(B)(Caly^2, S16^7, Workday^1, Employment^6,
Preccaly^3, Employee^5, S13A^4, Employer,
Service) :-
§3306(c)(Employee, Employer, Service).

Input 2 (§63(c)(5))

In the case of an individual^1 with respect to whom a deduction^2 under section 151 is allowable to another taxpayer^3 for a taxable year^4 beginning in the calendar year^5 in which the individual^1's taxable year^6 begins, the basic standard deduction^7 applicable to such individual^1 for such individual^1's taxable year^6 shall not exceed the greater^8 of-

$\{(15,27),(50,60),(96,111),(117,130),(145,161),(172,185),(189,200),(210,237),(253,267),(273,287),(291,302),(321,331)\}$

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Output 2

§63(c)(5)(Bassd^7, Grossinc, S45^3, Taxp^1, Taxy^6,
S44B^2, S46B^4, S47^5, S48^8) :-
[
§151(b)(Spouse=Taxp, Taxp=S45, Taxy) OR
§151(c)(S24A=Taxp, Taxp=S45, Taxy)
] AND
§63(c)(5)(A)() AND
§63(c)(5)(B)(Grossinc, Taxp).

Input 3 (§1(d)(iv))

(iv) \$31,172, plus 36% of the excess over \$115,000^1 if the taxable income^2 is over \$115,000 but not over \$250,000;
$\{(5,45),(50,67)\}$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Output 3

§1(d)(iv)(Tax^1, Taxinc^2).

A.4 Argument instantiation

The following are example cases. In addition to the case description, subsection to apply and input argument-value pairs, the agent has access to the output of Argument identification, Argument coreference and Structure extraction, for the entirety of the statutes.

Input 1: case 3306(a)(1)(B)-positive

Case description: Alice has employed Bob on various occasions during the year 2017: Jan 24, Feb 4, Mar 3, Mar 19, Apr 2, May 9, Oct 15, Oct 25, Nov 8, Nov 22, Dec 1, Dec 3.
Subsection to apply: §3306(a)(1)(B)

Argument-value pairs: {Employer="Alice", Caly="2017"}

Output 1

{Workday=["Jan 24", "Feb 4", "Mar 3", "Mar 19", "Apr 2", "May 9", "Oct 15", "Oct 25", "Nov 8", "Nov 22", "Dec 1", "Dec 3"], Employee="Bob", Employment="has employed", "S13A": [4, 5, 9, 11, 13, 19, 41, 43, 45, 47], @truth=True}

Input 2: case §63(c)(5)-negative

Case description: In 2017, Alice was paid $33200. Alice and Bob have been married since Feb 3rd, 2017. Bob earned $10 in 2017. Alice and Bob file separate returns. Alice is not entitled to a deduction for Bob under section 151.

Subsection to apply: §63(c)(5)

Argument-value pairs: {Taxp="Bob", Taxy="2017", Bassd=500}

Output 2

{@truth=False}

Input 3: tax case 5

Case description: In 2017, Alice's gross income was $326332. Alice and Bob have been married since Feb 3rd, 2017, and have had the same principal place of abode since 2015. Alice was born March 2nd, 1950 and Bob was born March 3rd, 1955. Alice and Bob file separately in 2017. Bob has no gross income that year. Alice takes the standard deduction.

Subsection to apply: Tax

Argument-value pairs: {Taxy="2017", Taxp="Alice"}

Output 3

{Tax=116066, @truth=True}

B Dataset statistics

B.1 Argument identification

Table 6 reports statistics on the annotations for the argument identification task. The numbers in that table were used to plot the top histogram in Figure 2(a).

Counts	SARA	Random
0	33	24
1	39	16
2	34	21
3	32	21
4	13	11
5	11	12
6	7	13
7	4	6
8	10	4
9	5	1
10	2	7
11	2	3
12	1	3
13	0	2
14	1	0
15	0	0
16	0	2
total	194	146
Statistics		
average	3.0	4.0
stddev	2.8	3.6
median	2	3

Table 6: Number of argument placeholders per subsection. "Counts" reports the number of subsections (right columns) containing a specific number of placeholders (left column). "Random" refers to 9 sections drawn at random from the Tax Code, and annotated.

B.2 Argument coreference

In Tables 7 and 8, we report statistics on the annotations for the argument coreference task. The numbers in Table 7 (resp. 8) were used to plot the middle (resp. bottom) histogram in Figure 2(a).

Counts	SARA	Random
0	33	24
1	40	22
2	44	18
3	30	23
4	15	18
5	10	14
6	13	13
7	5	8
8	4	2
9	0	3
10	0	1
total	161	146
Statistics		
average	2.4	3.1
stddev	2.0	2.4
median	2	3

Table 7: Number of arguments per subsection. "Counts" reports the number of subsections (right columns) containing a specific number of arguments (left column). "Random" refers to 9 sections drawn at random from the Tax Code, and annotated.

Counts	SARA	Random
1	391	360
2	70	73
3	6	16
4	6	6
5	0	1
total	473	456
Statistics		
average	1.2	1.3
stddev	0.5	0.6
median	1	1

Table 8: Number of mentions per argument. "Counts" reports the number of arguments (right columns) mentioned a specific number of times (left column). "Random" refers to 9 sections drawn at random from the Tax Code, and annotated.

B.3 Structure identification

Table 9 reports statistics on the annotations for the structure extraction task. These numbers for arguments differ from those in Table 6, because any subsection is allowed to contain the arguments of any subsections it refers to.

Counts	Arguments	Dependencies
0	9	80
1	13	42
2	40	28
3	60	18
4	24	8
5	13	2
6	14	3
7	7	7
8	7	1
9	5	1
10	-	0
11	-	0
12	-	2
total	192	192
Statistics		
average	3.0	1.0
stddev	2.6	2.4
median	3	1

Table 9: Number of arguments and dependencies of each subsection, as represented in the structure annotations. "Counts" reports the number of arguments or dependencies (right columns) mentioned a specific number of times (left column).

B.4 Argument instantiation

Tables 10 and 11 show statistics for the annotations for the argument instantiation task. In the gold data, we separate training and test data, to show that both distributions are close.

		Gold		Silver
Counts	train	test	all	
0	7	8	15	1197
1	24	13	37	5487
2	177	73	250	35629
3	41	24	65	32751
4	5	2	7	447
5	2	0	2	32
total	256	120	376	75543
Statistics				
average	2.1	2.0	2.0	2.3
stddev	0.7	0.8	0.7	0.7
median	2	1	2	2

Table 10: Number of arguments-value pairs for the input to the argument instantiation task. "Counts" reports the number of arguments (right columns) mentioned a specific number of times (left column). "Gold" refers to the manually annotated data, and "Silver" to the data produced automatically through the Prolog program.

		Gold		Silver
Counts	train	test	all	
1	131	78	209	41248
2	96	33	129	17051
3	12	4	16	8712
4	7	3	10	6656
5	8	2	10	1573
6	1	0	1	242
7	1	0	1	51
8	0	0		8
9	0	0		2
total	256	120	376	75543
Statistics				
average	1.7	1.5	1.6	1.8
stddev	1.0	0.8	1.0	1.1
median	1	1	1	1

Table 11: Number of arguments-value pairs for the output to the argument instantiation task. "Counts" reports the number of arguments (right columns) mentioned a specific number of times (left column). "Gold" refers to the manually annotated data, and "Silver" to the data produced automatically through the Prolog program.