Connecting Symbolic Statutory Reasoning with Legal Information Extraction

Nils Holzenberger
Télécom Paris
Institut Polytechnique de Paris
nils.holzenberger@telecom-paris.fr

Benjamin Van Durme
Johns Hopkins University
vandurme@jhu.edu

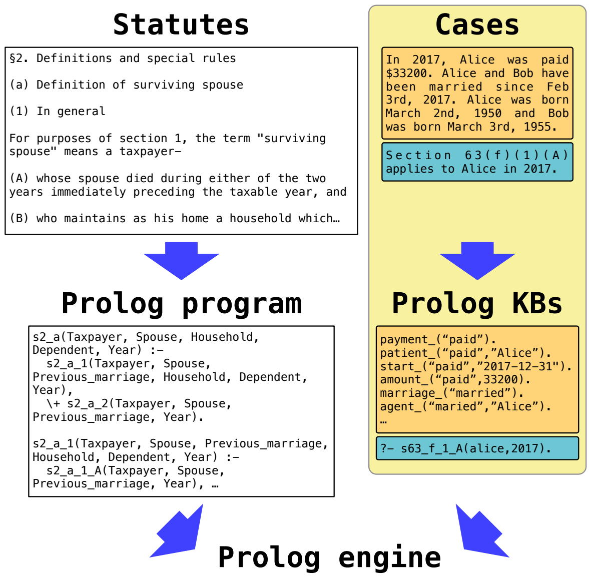
Abstract

Statutory reasoning is the task of determining whether a given law – a part of a statute – applies to a given legal case. Previous work has shown that structured, logical representations of laws and cases can be leveraged to solve statutory reasoning, including on the StAtutory Reasoning Assessment dataset (SARA), but rely on costly human translation into structured representations. Here, we investigate a form of legal information extraction atop the SARA cases, illustrating how the task can be done with high performance. Further, we show how the performance of downstream symbolic reasoning directly correlates with the quality of the information extraction.

1 Introduction

Statutory reasoning is the task of reasoning about legal cases with legal statutes. It is generally contrasted with case-based reasoning, and is complementary to it in legal systems based on common law (Lawsky, 2017). The StAtutory Reasoning Assessment dataset (SARA) is a benchmark for statutory reasoning for US federal tax law (Holzenberger et al., 2020).

In some settings, laws and regulations can be encoded into first-order logic programs to reason about cases (El Hamdani et al., 2021; Merigoux et al., 2021a; Bench-Capon et al., 1987). In Holzenberger et al. (2020), SARA statutes and cases were manually translated into Prolog code, solving statutory reasoning in that context, in a process depicted in Figure 1. Here, we take the standpoint advocated by Merigoux et al. (2021a), and start with the premise that a structured form of the statutes is available, while cases are stated in natural language. In practice, many governments across the world use expert systems to compute taxes owed by taxpayers, maintained by a dedicated team of experts (Oskamp and Lauritsen, 2002; Merigoux et al., 2021b). In contrast, translating cases into a



```
§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–

(A) whose spouse died during either of the two
years immediately preceding the taxable year, and

(B) who maintains as his home a household which…
```

In 2017, Alice was paid $33200. Alice and Bob have been married since Feb 3rd, 2017. Alice was born March 2nd, 1950 and Bob was born March 3rd, 1955.

Section 63(f)(1)(A) applies to Alice in 2017.

```
s2_a(Taxpayer, Spouse, Household,
Dependent, Year) :-
  s2_a_1(Taxpayer, Spouse,
Previous_marriage, Household, Dependent,
Year),
  \+ s2_a_2(Taxpayer, Spouse,
Previous_marriage, Year).

s2_a_1(Taxpayer, Spouse, Previous_marriage,
Household, Dependent, Year) :-
  s2_a_1_A(Taxpayer, Spouse,
Previous_marriage, Year), …
```

```
payment_(“paid”).
patient_(“paid”,”Alice”).
start_(“paid”,”2017-12-31").
amount_(“paid”,33200).
marriage_(“married”).
agent_(“maried”,”Alice”).
…
```

```
?- s63_f_1_A(alice,2017).
```

Figure 1: Statutory reasoning connects statutes and case descriptions. In previous work, both were manually translated into Prolog code to enable formal reasoning. Here, we illustrate the right portion of the figure: automated translation of cases into Prolog.

Knowledge Base (KB) requires a large amount of repetitive labor. Automating this translation task would be more scalable, and could take advantage of the high-quality expert system built on statutes.

In this work, we develop a form of Information Extraction (IE) for legal cases (Filtz et al., 2020), corresponding to the task highlighted on the right of Figure 1. We release a new version of the SARA dataset, with updated annotations for the Prolog KBs, and establish baseline results in extracting those KBs from case descriptions. This allows us to answer the following:

- What representation of Knowledge Base Elements (KBEs) in Prolog is appropriate for IE?
- How well can IE models perform on SARA?
- How does performance in IE translate to performance in statutory reasoning?

We report improved scores on statutory reasoning, showing the benefit of a structured Prolog representation for statutes. We further show an intuitive correlation in IE performance with performance in statutory reasoning. This represents an exciting outcome for IE practitioners: the SARA dataset can provide a clearly motivated downstream task that is directly impacted by improved IE.

2 Related work

Statutory reasoning While the present work is mainly concerned with IE, it is done for the purpose of statutory reasoning (Holzenberger et al., 2020). This legal NLP task (Chalkidis et al., 2022; Zhong et al., 2020) is close to that of legal entailment (Rabelo et al., 2022) and case outcome prediction (Branting et al., 2021; Luo et al., 2017). Large Language Models (LLMs) have recently set state-of-the-art on statutory reasoning (Blair-Stanek et al., 2023; Guha et al., 2023).

Legal-domain IE IE has been done extensively on contracts, as a potentially high-impact application (Chalkidis et al., 2018; Glaser et al., 2018; Liepiņa et al., 2020). The goal is to automate routine legal tasks, such as legal contract analysis (Hendrycks et al., 2021) and contract management (Schneider et al., 2022). IE has also been attempted on regulatory texts: Chalkidis et al. (2019) released a dataset of legal documents produced by the EU annotated with a set of legal terms. There are datasets for classifying paragraphs in policies (Bartolini et al., 2004), extracting subject-object relationships (Alohaly et al., 2018), and identifying cross-references (Boella et al., 2019). Closer to this paper, IE has been applied to case law. Yao et al. (2022) introduce a dataset of Chinese court cases annotated with a custom event ontology. Navas-Loro and Santos (2018) offer a review of event extraction in the legal domain, both annotation schemas and computational models, as well as annotations of European Court of Justice cases. Filtz et al. (2020) provide a corpus of events based on the decisions of the European Court of Human Rights, and compare statistical and rule-based models for event extraction.

Semantic representations This paper is concerned with the extraction of events and their arguments by mapping the semantic content of language into a structured format (Sundheim, 1992). There are numerous formalisms for that purpose, and this section is limited to a few examples. Blackburn and Bos (2005) describe the role of first-order logic in representing the semantics of natural language. The book reviews fundamental issues in semantic representations, with the goal of making inferences. Abstract Meaning Representation (AMR) (Banarescu et al., 2013) represents the semantic content of sentences, without alignment to the syntax of the source text. Universal Conceptual Cognitive Annotation (Hershcovich et al., 2017) is aimed at more flexibility than AMR, especially cross-linguisticality, and its graphs feature more challenges for parsing than AMR graphs. Universal Decompositional Semantics (White et al., 2020) provides multiple layers of annotations, separating syntactic and semantic information, and decompositional semantics. There are semantic representations that aim to reflect the full content of language, such as Episodic Logic (Kim et al., 2021; Schubert et al., 2010). Closer to this work, the PropBank dataset (Palmer et al., 2005; O'Gorman et al., 2018) adds predicate-argument relations on top of the Penn Treebank parse trees (Marcus et al., 1993). This effort has been extended to more languages (Moeller et al., 2020; Anwar et al., 2016; Wu and Palmer, 2015). The structures considered here are represented using Neo-Davidsonian semantics (Castañeda, 1967; Davidson, 1967; Parsons, 1990). In particular, they only capture the content relevant for interpreting the SARA statutes. Previous work on semantic representations for the legal domain, including the tax-law domain, have made use of ad-hoc ontologies, depending on the application domain (Bench-Capon et al., 1987; Sherman, 1987; Merigoux et al., 2021a).

Task-oriented representations When performing semantic parsing for a specific task, it is possible to sacrifice expressivity for performance (Kollar et al., 2018). Zelle and Mooney (1996) introduce CHILL, a shift-reduce parser producing queries executable against the Geoquery database. With the same benchmark dataset, Zettlemoyer and Collins (2005) induce a grammar and a log-linear model to map questions to queries. Similarly, Berant et al. (2013) answer questions stated in natural language to a KB. They translate the question into a query that can be executed against the KB. Beltagy et al. (2014) use semantic parsing for the tasks of Recognizing Textual Entailment and Semantic Textual Similarity. Zhang et al. (2019) and Weir et al. (2020) generate SQL queries from natural language questions. Here too, compositionality can play a role (Pasupat and Liang, 2015). Campagna et al. (2019) focus on parsing language for virtual assistant commands. Goldwasser et al. (2011) introduce a method to train a semantic parser with self-supervision, using a model's own high-confidence predictions on the Geoquery dataset as training examples. Liang et al. (2011) offer an alternate approach, treating the semantic parse as a latent variable. We are not aware of any prior work on legal-domain task-oriented semantic parsing.

3 Annotation of SARA

In Holzenberger et al. (2020) and again in Holzenberger and Van Durme (2021), the semantics of each SARA case have been translated into a set of KBEs. These annotations, based on a custom ontology in Prolog, effectively make it possible to solve SARA with a Prolog program that reflects the semantics of the statutes. However, with IE in mind, the ontology has several design flaws. We will focus on two of them. First, the events are represented using strings and integers, without clear reference to the text of the case description. Second, the use of the ontology is somewhat inconsistent. We improve over this annotation scheme to enable the training of IE models.[1]

3.1 SARA semantics

The existing ontology relies on a set of predefined predicates, denoted using strings ending in "_". Those roughly 60 predicates are used to specify event and argument types. Instances of entities and events are referred to using strings and integers. For example, line 1 in Figure 2 expresses that `"gross income"` is an event of type `income_`.[2] Line 2 expresses that the event `"gross income"` occurred on January 1st, 2017. Line 3 expresses that the entity `"Alice"` is the `agent_` of the event `"gross income"`. Predicates that specify event types take a single argument, which is an instance of an event. Predicates that specify arguments take two arguments: the first one is the instance of an event, and the second one is an entity or, occasionally, an event. This follows Neo-Davidsonian semantics (Davidson, 1967).

Some case descriptions state facts that are repeated across multiple instances — see Appendix A for an example. Some cases explicitly state that a given part of the SARA statutes applies: "Alice's taxable income for the year 2017 is $22895." (case `s1_c_iii_neg`). These are expressed in terms of the corresponding Prolog predicates: `s63("Alice",2017,22895,_).`

3.2 Design choices

Events and entities Our main improvement over Holzenberger and Van Durme (2021) is that we represent entities and events with a direct reference to the text of the case description. This is common practice in IE annotations (Walker et al., 2006). For that, entities and events are represented as structured `span` objects: `span(Value, Start_index, End_index).`

`Start_index` and `End_index` are inclusive character indices into the case description, providing an anchor in the text for the entity or event. `Value` is a string or integer meant to be used internally by the Prolog program. For events and people, this is a string, which is often the same as the anchor, e.g. `"Alice"`. As much as possible, we use single words to represent events, as can be seen in the bottom left of Figure 4. For dates, `Value` is an integer representation of the date, as in lines 2 and 8 of Figure 2. `Value` can generally be parsed out of the anchor using rule-based heuristics (see Appendix B). For dollar amounts, `Value` is an integer (inferable from the anchor), as on line 4.

Ontology We enforce constraints on the ontology to make it more consistent, and thus improve the training of the IE system. Events that are considered instantaneous for the purposes of statutory reasoning are allowed to have a `start_` argument, but no `end_` argument.[3] See e.g. lines 17 and 21 in Figure 2, which are present in Holzenberger and Van Durme (2021), but absent in our annotation. Whenever the day of an event is not specified but only the year, we enforce that the value for `start_` is the first day of the specified year, and the last day for `end_` — see e.g. line 2.

Prolog program We adjust the Prolog program to the changes mentioned above. We implement functions to extract year, month and day from the integer representations of dates. When comparing

[1]The data can be found at https://nlp.jhu.edu/law/sara_v3.

[2]We use `Prolog` syntax highlighting, and occasionally omit final periods, which are normally part of Prolog syntax.

[3]Those are `birth_`, `death_`, `deduction_`, `income_`, `joint_return_` and `payment_`.

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.

1	`income_("gross income")`	`income_(span("income", 23, 28))`
2	`start_("gross income", "2017-12-31")`	`start_(span("income", 23, 28), span(20170101, 3, 6))`
3	`agent_("gross income", "Alice")`	`agent_(span("income", 23, 28), span("Alice", 9, 13))`
4	`amount_("gross income", 326332)`	`amount_(span("income", 23, 28), span(326332, 35, 40))`
5	`marriage_("married")`	`marriage_(span("married", 67, 73))`
6	`agent_("married", "Alice")`	`agent_(span("married", 67, 73), span("Alice", 43, 47))`
7	`agent_("married", "Bob")`	`agent_(span("married", 67, 73), span("Bob", 53, 55))`
8	`start_("married", "2017-02-03")`	`start_(span("married", 67, 73), span(20170203, 81, 93))`
9	`residence_("principal place of abode")`	`residence_(span("abode", 137, 141))`
10	`agent_("principal place of abode", "Alice")`	`agent_(span("abode", 137, 141), span("Alice", 43, 47))`
11	`agent_("principal place of abode", "Bob")`	`agent_(span("abode", 137, 141), span("Bob", 53, 55))`
12	`patient_("principal place of abode", "the same principal place of abode")`	`patient_(span("abode", 137, 141), span("place", 128, 132))`
13	`start_("principal place of abode", "2015-01-01")`	`start_(span("abode", 137, 141), span(20150101, 149, 152))`
14	`birth_("Alice was born")`	`birth_(span("born", 165, 168))`
15	`agent_("Alice was born", "Alice")`	`agent_(span("born", 165, 168), span("Alice", 155, 159))`
16	`start_("Alice was born", "1950-03-02")`	`start_(span("born", 165, 168), span(19500302, 170, 184))`
17	`end_("Alice was born", "1950-03-02")`	[REMOVED]
18	`birth_("Bob was born")`	`birth_(span("born", 198, 201))`
19	`agent_("Bob was born", "Bob")`	`agent_(span("born", 198, 201), span("Bob", 190, 192))`
20	`start_("Bob was born", "1955-03-03")`	`start_(span("born", 198, 201), span(19550303, 203, 217))`
21	`end_("Bob was born", "1955-03-03")`	[REMOVED]

Figure 2: Case description and semantic annotation of `tax_case_5`. Left column: annotations from Holzenberger and Van Durme (2021). Right column: updated annotations. KBEs are aligned. The term [REMOVED] indicates content deprecated under the new standard introduced here.

events, we compare value, start and end indices. This enables distinguishing the events on lines 14 and 18 of Figure 2, even though the textual form of their anchor is identical. When comparing entities, we only compare values, enabling basic coreference, as between the `agent_` on lines 3 and 15.

Content of cases Further, we modified 4 cases. The conclusion of `s152_a_neg` was ambiguous, and possibly wrong, because Bob did indeed fulfill the criteria to be a dependent of Alice, as a qualifying child. We found that because of Prolog's closed-world assumption, Alice was triggering an exception, causing the Prolog solver to produce the expected (but incorrect) conclusion. We minimally edited the case to disambiguate it and produce the expected negative conclusion. We found cases `s152_b_1_pos` and `s152_b_1_neg` to be somewhat ambiguous, and added a single disambiguating sentence. Finally, throughout the world, Sunday is alternatively considered to be either the first or the last day of the week. This has an impact on cases involving section 3306. To avoid any ambiguities, in `3306_a_1_B_neg`, we changed March 19, 2017 to March 18; April 2, 2017 to April 1; and December 3, 2017 to December 2.

4 Model

Overview We frame the problem of IE as mapping the case description (a string) to a tree of depth 2 or less, like the one on Figure 3. Each node of the tree is a span in the case description, and each edge is labeled with an element from the ontology. The tree is built incrementally from the root: for each node, we predict its child nodes and the corresponding labelled edges. The example in Figure 3 is predicted in 3 steps, predicting (1) the nodes with full lines, (2) the top 3 nodes with dashed lines, and (3) the bottom 3 nodes with dashed lines. This section details the steps in this procedure, and line numbers refer to Algorithm 1, which describes how individual nodes and edges are predicted. Algorithm 2 describes how trees are predicted. To put this approach into context: we use a span-based parser, which incrementally segments the input text and types each segment (Das et al., 2014). We use the model described in Section 4.5 of Yarmohammadi et al. (2021), which has state-of-the-art performance for FrameNet parsing (Xia et al., 2021). We make one major change: we use a multi-layer perceptron instead of a BiLSTM. Appendix C describes how predicted KBEs are associated with probabilities.

```
Alice and Bob got married on April 5th, 2012. Alice and Bob were legally
separated under a decree of divorce on September 16th, 2017.
```

```
marriage_(span("married", 18, 24)).
agent_(span("married", 18, 24), span("Alice", 0, 4)).
agent_(span("married", 18, 24), span("Bob", 10, 12)).
start_(span("married", 18, 24), span(20120405, 29, 43)).
legal_separation_(span("separated", 73, 81)).
patient_(span("separated", 73, 81), span("married", 18, 24)).
agent_(span("separated", 73, 81), span("decree of divorce", 91, 107)).
start_(span("separated", 73, 81), span(20170916, 112, 131)).
```

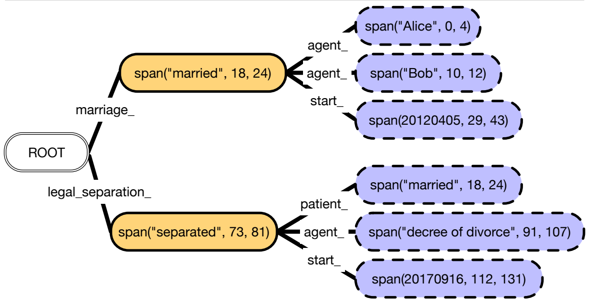


Figure 3: Inputs and outputs of the IE task for case `s7703_a_2`. Top frame: input case description. Bottom frame: output KB. Bottom: tree representation of the KB. Events are in orange with full line, and their arguments in purple with dashed line.

Algorithm 1: Find the children of a given node and labels of edges. FINDCHILDREN(X, r)

```
Require: text $X$, root span $r$
Ensure: knowledge base elements $K$
 1: $K \leftarrow \emptyset$
 2: $H \leftarrow$ EMBED$(X)$   # embed text with pre-trained encoder $\mathcal{E}$
 3: $F \leftarrow$ AUGMENT$(H, r)$   # add distance features
 4: $x \leftarrow$ COMPUTESPANREPRESENTATION$(H, r)$   # compute contextualized representation of root span $r$
 5: $S \leftarrow$ BIOTAGGER$(F, x)$   # extract spans with BIO tagger $\mathcal{T}$
 6: for $s$ in $S$ do
 7:     $z \leftarrow$ COMPUTESPANREPRESENTATION$(F, s)$   # compute contextualized representation of span $s$
 8:     $l \leftarrow$ PREDICTLABEL$(z, x)$   # predict label with classifier $\mathcal{L}$
 9:     $K \leftarrow K \cup \{(l, r, s)\}$   # add new KBE
10: end for
11: return $K$
```

Algorithm 2: Predict the IE tree for a given case. PREDICTTREE(X)

```
Require: case description $X$
Ensure: tree $T$
 1: $T \leftarrow \emptyset$
 2: $Q \leftarrow$ [ROOT]   # queue of nodes to be processed
 3: while $Q$ is not empty do
 4:     $q \leftarrow$ POP$(Q)$   # take first node
 5:     $K \leftarrow$ FINDCHILDREN$(X, q)$   # find its children
 6:     $T \leftarrow T \cup K$   # add predicted relations to tree
 7:     if $q$ is ROOT then
 8:         for $l, r, s \in K$ do
 9:             $Q \leftarrow Q \cup \{s\}$   # add children of the root to the queue, to predict one more level of children
10:         end for
11:     end if
12: end while
13: return $T$
```

Features We use LLMs as pre-trained encoders (see Section 5.2). The case description is tokenized into $X = x_1, x_2, ..., x_n$ using the LLM's tokenizer. X is mapped to features $H = h_1, h_2, ..., h_n$ using the pre-trained encoder (line 2). Given the span of the parent node, we augment h_i with the distance between the span of the parent node and i. This distance is expressed in number of tokens, and represented using learned embeddings. If the parent node is the root of the tree, then we set the distance feature to the zero vector. We concatenate h_i and the distance feature p_i to obtain $f_i = [h_i, p_i]$ $(1 \leq i \leq n)$ where $[.,.]$ denotes concatenation (line 3). Further, we compute x, the representation of the parent node, as the average of the tokens that represent the span of the parent node (line 4). If the parent node is the root, we set $x = v$ where v is a learnable vector embedding.

Finding nodes Given the parent node, we find its children as spans in the case description. We map each $g_i = [f_i, x]$ $(1 \leq i \leq n)$ to BIO logits with a multi-layer perceptron. We use a Conditional Random Field (Lafferty et al., 2001) with BIO tags to predict a set of spans $S = \{s_1, s_2, ..., s_m\}$. We denote this BIO tagger as $\mathcal{T}$ (line 5).

Predicting labels We now label the edges connecting each child node to the parent. Each span $s_j \in S$ has a start index s_j^{start} and an end index s_j^{end}.[4] On line 7, we map each span s_j to its representation $q_j = \frac{1}{s_j^{\text{end}} - s_j^{\text{start}} + 1} \sum_{k=s_j^{\text{start}}}^{s_j^{\text{end}}} f_k$ i.e. the average of the features in s_j.[5] We use a multi-layer perceptron $\mathcal{L}$ to map each $u_j = [q_j, x]$ $(1 \leq j \leq m)$ to label logits l_j (line 8).

Training Our model consists of encoder $\mathcal{E}$, BIO tagger $\mathcal{T}$, label classifier $\mathcal{L}$, distance embeddings $\mathcal{D}$, and root representation v. We have gold spans for all nodes in the tree, that we use to generate gold sequences of BIO tags and to train $\mathcal{T}$. We

[4] At training time, S is the set of gold spans. At eval time, S is the set of spans predicted by the BIO tagger.

[5] For roughly 80% of gold spans, the span is a single token, so that $s_j^{\text{start}} = s_j^{\text{end}}$, and this amounts to selecting a single vector in the sequence $f_1, f_2, ..., f_n$.

use the loss

$$L_{\mathcal{T}} = -\frac{1}{|G|} \log p(G)$$

where G is the gold sequence of BIO tags, and $|G|$ its length in tokens. The labels of the edges are used to train $\mathcal{L}$, with the loss

$$L_{\mathcal{L}} = -\frac{1}{|U|} \sum_{(l,r,c) \in U} \log p(l|r, c)$$

where (l, r, c) is a triplet of (label, parent node, child node) from the training set U. Finally, $\mathcal{E}$, $\mathcal{D}$ and v are used to compute features input to $\mathcal{T}$ and $\mathcal{L}$. Information is passed from one module to the next in a differentiable way, making the entire model trainable end-to-end with the single loss function

$$L = \lambda L_{\mathcal{T}} + (1 - \lambda) L_{\mathcal{L}}$$

where $\lambda \in [0, 1]$ is a tradeoff hyperparameter.

5 Experiments

5.1 Data pre-processing

First, we run Prolog queries against each case to produce an exhaustive inventory of events and arguments. For example, the query `?- income_(X).` returns all events of type `income_`, and the query `?- agent_(X, Y).` returns all `X, Y` pairs such that entity `Y` is the agent of event `X`. We thus produce a set of cases containing only KBEs, to avoid the need to extract the rules shown in Appendix A. This dataset we refer to as the *full dataset*.

Some spans cannot be extracted using the IE framework described in Section 4. First, it cannot deal with overlapping spans. Out of 2 overlapping spans we keep the one that appears earliest in the case description. Second, some values cannot be inferred from their anchor. For example, some sentences describe repeating events, without explicitly mentioning all the dates of occurrence. This makes it impossible to extract those unmentioned dates. Appendix A describes one such event.

We produce the *partial dataset* by removing all unextractable spans and corresponding KBEs. Table 1 shows statistics on how many KBEs were removed in the process. The number of KBEs rejected can seem large in comparison to those kept. However, a fraction of the cases contains most rejected KBEs. Ranking the cases by number of KBEs rejected, the top 10% represent 96%, 98% and 71% of the rejected KBEs in train, dev and test, respectively. Figure 4 in Appendix G shows statistics on the partial dataset.

At training time, we use the partial dataset. At test time, we report IE scores against both the partial (Table 2) and full datasets (Table 4). To clarify the difference: the partial dataset contains only those KBEs from the full dataset that our framework is designed to handle. Both contain the same SARA cases. Note that this filtering only mildly affects the overall task of statutory reasoning, as can be seen in Table 3, TOPLINE: more than 90% of the cases in the partial dataset can still be solved by the Prolog program, despite missing KBEs.

	Train	Dev	Test
Kept	2830	664	1305
Rejected	3099	411	1719

Table 1: Number of KBEs kept and rejected, as described in Section 5.1.

The SARA dataset only provides a train/test split. For the purposes of early stopping and hyperparameter search, we split up the train set into a train and dev set, the same way that the train/test split was created in Holzenberger et al. (2020). We pair binary cases by the part of the statutes that they test: e.g. `s2_b_1_pos` and `s2_b_1_neg` are grouped together. We randomly select 20% of these groups for the dev set. We also randomly pick 20% of numerical cases to be used in the dev set. The result is a train set with 206 cases and a dev set with 50 cases.

5.2 Results

Information Extraction We use the AllenNLP codebase (Gardner et al., 2018) version 2.7.0 to code and train models.[6] We use stochastic gradient descent with early stopping on the dev set.

Performance of a given model is measured as its F1 score, averaged across cases. We count a KBE predicted by the model as correct if it appears exactly in the gold KB. The model receives no credit for a slightly inaccurate span or label.

We use Optuna (Akiba et al., 2019) to search for the best set of hyperparameters. Appendix F reports the range of hyperparameters explored and best hyperparameters found. We measure performance for a given set of hyperparameters as the

[6]Code for experiments can be found at https://github.com/SgfdDttt/sara-ie.

mean F1 score across 3 different runs, with 3 different random seeds. This is to mitigate variability from random initializations.

In Table 2, we report precision, recall and F1 scores, for dev and test sets, on the partial dataset. We report results on the full dataset in Table 4 in Appendix D. We compare 5 different pre-trained encoders (used to map X to H in Section 4):

- LEGALBERT: the legal-domain encoder of Holzenberger et al. (2020), trained on a subset of the `case.law` corpus (caselaw), starting from `bert-base-cased`,
- LEGALBERT': an alternative trained by Zheng et al. (2021) on more data,
- BERT: BERT-base-cased (Devlin et al., 2019), which has the same architecture and training objective as both LEGALBERTs, but was trained on general-domain English,
- ROBERTA: RoBERTa-base (Liu et al., 2019) which was shown to achieve higher performance than BERT on downstream tasks, and
- T5: T5-base (Raffel et al., 2020), which was shown to achieve higher performance than RoBERTa on downstream tasks. We only use the encoder of T5-base in our experiments, discarding the decoder.

Error analysis We compare model outputs on 3 binary cases and 3 numerical cases picked at random from the test set.[7] We report the detail of the error analysis in Appendix E. From this sample, it seems that models are generally able to correctly detect all event anchors, and to further type them correctly in at least 90% of cases. Predicting arguments correctly is still a challenge, and an important one: a single missing argument can throw off the Prolog program, especially in computing tax amounts. For example, missing the `amount_` argument to an `income_` event is practically as bad as not detecting the `income_` event in the first place. When arguments are predicted, their value tends to be correct, and in any case of the right type (strings *versus* integers). For example, models rarely mistake the `agent_` of an event: they either predict it correctly, or miss it entirely. In addition, the more events there are, the harder it is to get all arguments right. To summarize, models mainly struggle with arguments. They both (1) miss arguments entirely, having generally higher precision than recall, and (2) occasionally predict additional arguments, such as predicting two `amount_` arguments by tagging all integers preceded by a "$" sign.

[7]In binary cases, the outcome to be predicted is either "Yes" or "No". In numerical cases, the outcome is an integer, representing a dollar amount of taxes owed.

Statutory reasoning We further measure how useful model outputs are for statutory reasoning, and report results in Table 3. We feed the KB produced with IE to the Prolog program. We add any statements about statutes of the form `s63("Alice",2017,22895,_)` to the KB, as described in Section 3.1. In addition to the five models above, we report results on (1) TOPLINE, which is the partial KB used to train models, and (2) EMPTY, which corresponds to a model that predicts no KBEs. Since we evaluated each model by running training and test with 5 different random seeds, we pick the model with the highest dev F1 score to produce the KB used for this evaluation on statutory reasoning.

5.3 Discussion

Focusing on the IE scores, LEGALBERT performs best out of all encoders, and T5 performs worst, with large performance gaps between encoders. For T5, only the encoder was used, and presumably, more training data is needed to retrain it as a standalone encoder. Legal-domain pre-training clearly yields performance improvements. As can be seen by comparing LEGALBERT and LEGALBERT', for the SARA dataset, more data does not automatically translate to better performance.

As noted in the caption of Table 3, results from Holzenberger et al. (2020), Holzenberger and Van Durme (2021) and Blair-Stanek et al. (2023) are not directly comparable to ours. Holzenberger et al. (2020) and Blair-Stanek et al. (2023) apply NLP models directly to the text of the cases and statutes, without using any Prolog code. Holzenberger and Van Durme (2021) rely on a set of lightweight semantic annotations of the statutes and cases to develop a neuro-symbolic model for statutory reasoning; they also exclude Prolog code. Contrary to prior work mentioned in this paragraph, we use a human-curated Prolog representation of the statutes.

Holzenberger et al. (2020) report that majority baselines achieve 50% entailment, and 20% numerical accuracies on the test set. Performance of

Model	Dev Precision	Recall	F1	Test Precision	Recall	F1
LEGALBERT	**88.7** ± 0.6	**88.6** ± 0.6	**88.1** ± 0.3	**88.1** ± 0.2	**91.0** ± 0.5	**87.0** ± 0.3
LEGALBERT'	87.4 ± 1.0	83.2 ± 1.4	84.4 ± 0.8	85.4 ± 0.9	84.6 ± 1.4	82.7 ± 0.9
BERT	85.9 ± 2.1	75.1 ± 2.5	78.7 ± 1.6	85.2 ± 2.3	77.0 ± 2.3	77.4 ± 1.8
ROBERTA	84.5 ± 0.6	80.8 ± 1.5	81.9 ± 0.9	86.4 ± 0.7	83.9 ± 1.6	82.0 ± 1.0
T5	73.2 ± 0.6	56.2 ± 2.4	62.1 ± 1.5	71.6 ± 1.1	59.0 ± 1.3	61.4 ± 1.0

Table 2: Performance metrics on the IE task measured on the partial dataset. Scores are computed across 5 runs with different random seeds. We format statistics as *average* ± *standard error*. Best scores are **bolded**.

Model	Entailment Train	Dev	Test	Numerical Train	Dev	Test
Holzenberger et al. (2020)	-	-	55 ± 8.3	-	-	25 ± 17.1
Holzenberger and Van Durme (2021)	-	-	59 ± 8.2	-	-	45 ± 19.7
Blair-Stanek et al. (2023)	-	-	71 ± 7.6	-	-	-
TOPLINE	93.7 ± 3.4	97.1 ± 5.0	97 ± 2.8	98.4 ± 2.6	93.8 ± 11.0	90 ± 11.9
EMPTY	69.7 ± 6.4	67.6 ± 13.8	71 ± 7.6	26.6 ± 9.3	18.8 ± 17.7	25 ± 17.2
LEGALBERT	**88.7** ± 4.4	88.2 ± 9.5	**89** ± 5.2	**73.4** ± 9.3	**56.2** ± 22.5	**60** ± 19.4
LEGALBERT'	86.6 ± 4.7	**91.2** ± 8.4	86 ± 5.8	68.8 ± 9.7	50.0 ± 22.6	**60** ± 19.4
BERT	79.6 ± 5.6	73.5 ± 13.0	81 ± 6.5	46.9 ± 10.5	37.5 ± 21.9	45 ± 19.7
ROBERTA	79.6 ± 5.6	79.4 ± 11.9	79 ± 6.8	50.0 ± 10.5	43.8 ± 22.5	40 ± 19.4
T5	78.2 ± 5.8	73.5 ± 13.0	76 ± 7.1	31.2 ± 9.7	25.0 ± 19.6	25 ± 17.2

Table 3: Statutory reasoning accuracies (in %) with 90% confidence interval, obtained by applying the Prolog solver to the predicted KBs. Accuracies are measured using the metrics described in Holzenberger et al. (2020); numerical predictions need not match the ground truth exactly to be correct. Best scores within the IE-based approaches are **bolded**. The top three lines are not directly comparable, and are provided for reference.

EMPTY on statutory reasoning is higher than that. First, with the EMPTY KBs containing no facts, it is generally hard to prove anything at all, so that negative cases tend to succeed thanks to Prolog's negation as failure. Second, since statements about statutes are added automatically to the KB, cases that rely on them have a chance to succeed.

There is overall strong Pearson correlation between F1 and statutory reasoning scores: on the test set, 80% for entailment accuracy, and 88% for numerical accuracy. This shows that, for the SARA dataset, improvements in IE translate to improvements in statutory reasoning. This suggests that IE is important for statutory reasoning, and encourages progress on this task. IE models tend to perform better than EMPTY on statutory reasoning, so that even imperfect IE is useful for statutory reasoning. Still, there is room for improvement, both for IE and for statutory reasoning.

Here, the choice of ontology limits the information that the IE model can effectively extract from the SARA cases. In particular, the current ontology expresses time with two relations, `start_` and `end_`. In contrast, natural language can express time in many different ways, which can occasionally be hard to parse into a structured form (Filtz et al., 2020). SARA cases may contain the information that an event is ongoing, or happening at a certain point in time, without explicitly referring to a start or end time. Event calculus (Kowalski and Sergot, 1986; Miller and Shanahan, 2002) could be used to represent a wider spectrum of time expressions, reducing the amount of rejected KBEs. It would also make it possible for the logical representation to align more closely to the language of the case description. This would yield a better inductive bias for the IE models while not sacrificing expressivity. More generally, progress on IE could be made by picking a more expressive, more general ontology, such that the translation between language and logical form would be more straightforward. But this would require re-writing the Prolog solver, to accommodate the new ontology, and to encode relevant commonsense knowledge.

Finally, the use of explicit KBEs introduces interpretability and auditability in the model's output, since it becomes clear what facts are used to perform statutory reasoning. This is generally absent from the direct application of LLMs to statutory reasoning, even if this can be mitigated with prompt engineering (Blair-Stanek et al., 2023).

6 Conclusion

We improve the SARA ontology and re-annotate cases for their semantics. Compared to the previous iteration of the SARA dataset, the KBEs and overall structure are now richer and more complete. We also take a step towards more consistent annotations, and more extractable KBEs. We adapt state-of-the-art IE models to extract KBEs from case descriptions. With empirical results on the SARA dataset, we find that good performance with a strict metric can be reached on this small dataset, and that performance in IE translates directly into performance in statutory reasoning.

We see three main directions for future work. First, as stated in Section 5.1, the current ontology is not flexible enough to both (1) allow for KBEs closely aligned to the case description, and (2) express complex relationships, in particular time relationships. This is one major limitation that could be addressed, and would likely require overhauling the expert system representing the statutes. Second, we have so far assumed that the statutes were parsed into a logical form ahead of time. A meaningful step would be to augment the approach presented here with the automated extraction of the logical form of statutes, effectively performing semantic parsing. Finally, cases in SARA are not representative of the full scope and diversity of legal cases. Real-world legal cases are generally much longer, the language is denser, and the phrasing is more diverse. We see the choice of an ontology and the annotation of data as two crucial challenges.

7 Limitations

In this paper, we performed experiments on the SARA dataset, a semi-synthetic dataset for US federal tax law. The conclusions drawn about legal-domain IE and about statutory reasoning should take care to understand that this dataset is not representative of the full scope of legal data.

Acknowledgments

This work has been supported by the U.S. National Science Foundation under grant No. 2204926. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 2623–2631. ACM.

Manar Alohaly, Hassan Takabi, and Eduardo Blanco. 2018. A deep learning approach for extracting attributes of ABAC policies. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies, SACMAT 2018, Indianapolis, IN, USA, June 13-15, 2018*, pages 137–148. ACM.

Maaz Anwar, Riyaz Ahmad Bhat, Dipti Misra Sharma, Ashwini Vaidya, Martha Palmer, and Tafseer Ahmed Khan. 2016. A proposition bank of urdu. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA).

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013, August 8-9, 2013, Sofia, Bulgaria*, pages 178–186. The Association for Computer Linguistics.

Roberto Bartolini, Alessandro Lenci, Simonetta Montemagni, Vito Pirrelli, and Claudia Soria. 2004. Semantic mark-up of italian legal texts through nlp-based techniques. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*. European Language Resources Association.

Islam Beltagy, Stephen Roller, Gemma Boleda, Katrin Erk, and Raymond J. Mooney. 2014. Utexas: Natural language semantics using distributional semantics and probabilistic logic. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014*, pages 796–801. The Association for Computer Linguistics.

Trevor J. M. Bench-Capon, G. O. Robinson, Tom Routen, and Marek J. Sergot. 1987. Logic programming for large scale applications in law: A formalisation of supplementary benefit legislation. In *Proceedings of the First International Conference on Artificial Intelligence and Law, ICAIL '87, Boston, MA, USA, May 27-29, 1987*, pages 190–198. ACM.

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.

Patrick Blackburn and Johan Bos. 2005. *Representation and Inference for Natural Language - a First Course in Computational Semantics*. CSLI Studies in Computational Linguistics. CSLI Publications.

Andrew Blair-Stanek, Nils Holzenberger, and Benjamin Van Durme. 2023. Can GPT-3 perform statutory reasoning? *CoRR*, abs/2302.06100.

Guido Boella, Luigi Di Caro, and Valentina Leone. 2019. Semi-automatic knowledge population in a legal document management system. *Artif. Intell. Law*, 27(2):227–251.

Luther Karl Branting, Craig Pfeifer, Bradford Brown, Lisa Ferro, John S. Aberdeen, Brandy Weiss, Mark Pfaff, and Bill Liao. 2021. Scalable and explainable legal prediction. *Artif. Intell. Law*, 29(2):213–238.

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.

Hector Neri Castañeda. 1967. Comment on D. Davidson's "The logical forms of action sentences". *The Logic of Decision and Action*.

Ilias Chalkidis, Ion Androutsopoulos, and Achilleas Michos. 2018. Obligation and prohibition extraction using hierarchical rnns. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 254–259. Association for Computational Linguistics.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2019. 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.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael J. Bommarito II, Ion Androutsopoulos, Daniel Martin Katz, and Nikolaos Aletras. 2022. Lexglue: A benchmark dataset for legal language understanding in english. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 4310–4330. Association for Computational Linguistics.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Comput. Linguistics*, 40(1):9–56.

Donald Davidson. 1967. The logical forms of action sentences. *The Logic of Decision and Action*.

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.

Rajaa El Hamdani, Majd Mustapha, David Restrepo Amariles, Aurore Clément Troussel, Sébastien Meeùs, and Katsiaryna Krasnashchok. 2021. A combined rule-based and machine learning approach for automated GDPR compliance checking. In *ICAIL '21: Eighteenth International Conference for Artificial Intelligence and Law, São Paulo Brazil, June 21 - 25, 2021*, pages 40–49. ACM.

Erwin Filtz, María Navas-Loro, Cristiana Santos, Axel Polleres, and Sabrina Kirrane. 2020. Events matter: Extraction of events from court decisions. In *Legal Knowledge and Information Systems - JURIX 2020: The Thirty-third Annual Conference, Brno, Czech Republic, December 9-11, 2020*, volume 334 of *Frontiers in Artificial Intelligence and Applications*, pages 33–42. IOS Press.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Ingo Glaser, Bernhard Waltl, and Florian Matthes. 2018. Named entity recognition, extraction, and linking in german legal contracts. In *IRIS: Internationales Rechtsinformatik Symposium*, pages 325–334.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 1486–1495. The Association for Computer Linguistics.

Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, Jessica Wu, Joe Nudell, Joel

Niklaus, John J. Nay, Jonathan H. Choi, Kevin Tobia, Margaret Hagan, Megan Ma, Michael A. Livermore, Nikon Rasumov-Rahe, Nils Holzenberger, Noam Kolt, Peter Henderson, Sean Rehaag, Sharad Goel, Shang Gao, Spencer Williams, Sunny Gandhi, Tom Zur, Varun Iyer, and Zehua Li. 2023. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. *CoRR*, abs/2308.11462.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. CUAD: an expert-annotated NLP dataset for legal contract review. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1127–1138, Vancouver, Canada. Association for Computational Linguistics.

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.

Nils Holzenberger and Benjamin Van Durme. 2021. Factoring statutory reasoning as language understanding challenges. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2742–2758. Association for Computational Linguistics.

Gene Kim, Mandar Juvekar, and Lenhart Schubert. 2021. Monotonic inference for underspecified episodic logic. In *Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA)*, pages 26–40, Groningen, the Netherlands (online). Association for Computational Linguistics.

Thomas Kollar, Danielle Berry, Lauren Stuart, Karolina Owczarzak, Tagyoung Chung, Lambert Mathias, Michael Kayser, Bradford Snow, and Spyros Matsoukas. 2018. The alexa meaning representation language. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 3 (Industry Papers)*, pages 177–184. Association for Computational Linguistics.

Robert A. Kowalski and Marek J. Sergot. 1986. A logic-based calculus of events. *New Gener. Comput.*, 4(1):67–95.

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.

Sarah B Lawsky. 2017. A logic for statutes. *Fla. Tax Rev.*, 21:60.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 590–599. The Association for Computer Linguistics.

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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Bingfeng Luo, Yansong Feng, Jianbo Xu, Xiang Zhang, and Dongyan Zhao. 2017. Learning to predict charges for criminal cases with legal basis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2727–2736. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330.

Denis Merigoux, Nicolas Chataing, and Jonathan Protzenko. 2021a. Catala: a programming language for the law. *Proc. ACM Program. Lang.*, 5(ICFP):1–29.

Denis Merigoux, Raphaël Monat, and Jonathan Protzenko. 2021b. A modern compiler for the french tax code. In *CC '21: 30th ACM SIGPLAN International Conference on Compiler Construction, Virtual Event, Republic of Korea, March 2-3, 2021*, pages 71–82. ACM.

Rob Miller and Murray Shanahan. 2002. Some alternative formulations of the event calculus. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, volume 2408 of *Lecture Notes in Computer Science*, pages 452–490. Springer.

Sarah Moeller, Irina Wagner, Martha Palmer, Kathryn Conger, and Skatje Myers. 2020. The Russian PropBank. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5995–6002, Marseille, France. European Language Resources Association.

María Navas-Loro and Cristiana Santos. 2018. Events in the legal domain: first impressions. In *Proceedings of the 2nd Workshop on Technologies for Regulatory Compliance co-located with the 31st International Conference on Legal Knowledge and Information Systems (JURIX 2018), Groningen, The Netherlands, December 12, 2018*, volume 2309 of *CEUR Workshop Proceedings*, pages 45–57. CEUR-WS.org.

Tim O'Gorman, Sameer Pradhan, Martha Palmer, Julia Bonn, Kathryn Conger, and James Gung. 2018. The new propbank: Aligning propbank with AMR through POS unification. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Anja Oskamp and Marc Lauritsen. 2002. AI in law practice? so far, not much. *Artif. Intell. Law*, 10(4):227–236.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Terence Parsons. 1990. *Events in the Semantics of English*, volume 334. MIT press Cambridge, MA.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1470–1480. The Association for Computer Linguistics.

Juliano Rabelo, Randy Goebel, Mi-Young Kim, Yoshinobu Kano, Masaharu Yoshioka, and Ken Satoh. 2022. Overview and discussion of the competition on legal information extraction/entailment (COLIEE) 2021. *Rev. Socionetwork Strateg.*, 16(1):111–133.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Julián Moreno Schneider, Georg Rehm, Elena Montiel-Ponsoda, Víctor Rodríguez-Doncel, Patricia Martín-Chozas, María Navas-Loro, Martin Kaltenböck, Artem Revenko, Sotirios Karampatakis, Christian Sageder, Jorge Gracia, Filippo Maganza, Ilan Kernerman, Dorielle Lonke, Andis Lagzdins, Julia Bosque-Gil, Pieter Verhoeven, Elsa Gomez Diaz, and Pascual Boil Ballesteros. 2022. Lynx: A knowledge-based AI service platform for content processing, enrichment and analysis for the legal domain. *Inf. Syst.*, 106:101966.

Lenhart K. Schubert, Benjamin David Van Durme, and Marzieh Bazrafshan. 2010. Entailment inference in a natural logic-like general reasoner. In *Commonsense Knowledge, Papers from the 2010 AAAI Fall Symposium, Arlington, Virginia, USA, November 11-13, 2010*, volume FS-10-02 of *AAAI Technical Report*. AAAI.

David M. Sherman. 1987. A prolog model of the income tax act of canada. In *Proceedings of the First International Conference on Artificial Intelligence and Law, ICAIL '87, Boston, MA, USA, May 27-29, 1987*, pages 127–136. ACM.

Beth Sundheim. 1992. Overview of the fourth message understanding evaluation and conference. In *Proceedings of the 4th Conference on Message Understanding, MUC 1992, McLean, Virginia, USA, June 16-18, 1992*, pages 3–21. ACL.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus (LDC2006T06). *Philadelphia: Linguistic Data Consortium*.

Nathaniel Weir, Prasetya Utama, Alex Galakatos, Andrew Crotty, Amir Ilkhechi, Shekar Ramaswamy, Rohin Bhushan, Nadja Geisler, Benjamin Hättasch, Steffen Eger, Ugur Çetintemel, and Carsten Binnig. 2020. Dbpal: A fully pluggable NL2SQL training pipeline. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 2347–2361. ACM.

Aaron Steven White, Elias Stengel-Eskin, Siddharth Vashishtha, Venkata Subrahmanyan Govindarajan, Dee Ann Reisinger, Tim Vieira, Keisuke Sakaguchi, Sheng Zhang, Francis Ferraro, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2020. The universal decompositional semantics dataset and decomp toolkit. In *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, pages 5698–5707. European Language Resources Association.

Shumin Wu and Martha Palmer. 2015. Improving Chinese-English PropBank alignment. In *Proceedings of the Ninth Workshop on Syntax, Semantics* *and Structure in Statistical Translation*, pages 74–82, Denver, Colorado, USA. Association for Computational Linguistics.

Patrick Xia, Guanghui Qin, Siddharth Vashishtha, Yunmo Chen, Tongfei Chen, Chandler May, Craig Harman, Kyle Rawlins, Aaron Steven White, and Benjamin Van Durme. 2021. LOME: Large ontology multilingual extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 149–159, Online. Association for Computational Linguistics.

Feng Yao, Chaojun Xiao, Xiaozhi Wang, Zhiyuan Liu, Lei Hou, Cunchao Tu, Juanzi Li, Yun Liu, Weixing Shen, and Maosong Sun. 2022. LEVEN: A large-scale chinese legal event detection dataset. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 183–201. Association for Computational Linguistics.

Mahsa Yarmohammadi, Shijie Wu, Marc Marone, Haoran Xu, Seth Ebner, Guanghui Qin, Yunmo Chen, Jialiang Guo, Craig Harman, Kenton Murray, Aaron Steven White, Mark Dredze, and Benjamin Van Durme. 2021. Everything is all it takes: A multipronged strategy for zero-shot cross-lingual information extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 1950–1967. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2*, pages 1050–1055. AAAI Press / The MIT Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666. AUAI Press.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir R. Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 5337–5348. Association for Computational Linguistics.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. 2021. When does pretraining help?: assessing self-supervised learning for law and the casehold dataset of 53, 000+ legal holdings. In *ICAIL '21: Eighteenth International Conference for Artificial Intelligence and Law, São Paulo Brazil, June 21 - 25, 2021*, pages 159–168. ACM.

Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. 2020. How does NLP benefit legal system: A summary of legal artificial intelligence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5218–5230. Association for Computational Linguistics.

A Repeated events

The sentence "Bob earned $300000 every year from 2015 to 2019." (case `s2_b_3_B_pos`) is represented in Prolog as:

```
bob_income(Year, Event, Start_day,
           End_day) :-
    between(2015, 2019, Year),
    atom_concat("earned ", Year, Event),
    first_day_year(Year, Start_day),
    last_day_year(Year, End_day).

income_(span(Event, 286, 291)) :-
    bob_income(_, Event, _, _).

agent_(span(Event, 286, 291),
       span("Bob", 282, 284)) :-
    bob_income(_, Event, _, _).

amount_(span(Event, 286, 291),
        span(300000, 294, 299)) :-
    bob_income(_, Event, _, _).

start_(span(Event, 286, 291),
       span(Start_day, 317, 320)) :-
    bob_income(_, Event, Start_day, _).
```

This makes it possible to generate 4 different `payment_` events and their arguments:

```
income_(span("earned 2015", 286, 291)).
agent_(span("earned 2015", 286, 291),
       span("Bob", 282, 284)).
amount_(span("earned 2015", 286, 291),
        span(300000, 294, 299)).
start_(span("earned 2015", 286, 291),
       span(20150101, 317, 320)).

income_(span("earned 2016", 286, 291)).
agent_(span("earned 2016", 286, 291),
       span("Bob", 282, 284)).
amount_(span("earned 2016", 286, 291),
        span(300000, 294, 299)).
start_(span("earned 2016", 286, 291),
       span(20160101, 317, 320)).

income_(span("earned 2017", 286, 291)).
agent_(span("earned 2017", 286, 291),
        span("Bob", 282, 284)).
amount_(span("earned 2017", 286, 291),
         span(300000, 294, 299)).
start_(span("earned 2017", 286, 291),
        span(20170101, 317, 320)).

income_(span("earned 2018", 286, 291)).
agent_(span("earned 2018", 286, 291),
        span("Bob", 282, 284)).
amount_(span("earned 2018", 286, 291),
         span(300000, 294, 299)).
start_(span("earned 2018", 286, 291),
        span(20180101, 317, 320)).

income_(span("earned 2019", 286, 291)).
agent_(span("earned 2019", 286, 291),
        span("Bob", 282, 284)).
amount_(span("earned 2019", 286, 291),
         span(300000, 294, 299)).
start_(span("earned 2019", 286, 291),
        span(20190101, 317, 320)).
```

With the model of Section 4, it is not possible to generate a date in 2016, so that it cannot generate the fact:

```
start_(span("earned 2016", 286, 291),
        span(20160101, 317, 320))
```

B Dates in IE

B.1 Encoding

In order to facilitate handling time in Prolog, we represent a date as a base-10 integer of the form YYYYMMDD. For example, August 23rd, 2017 is represented as 20170823. In that, we follow Sherman (1987). This makes it straightforward to order two dates, to extract the day from date `d` as `d%100`, the month as `(d/100)%100`, and so on. It is also possible to compute the day of the week, or the index of the week in the current year. In practice, whenever the full date is stated in the case text, that is the anchor for the date. For instance, "Alice and Bob got married on August 24th, 1970" contains a clear anchor for the date 19700824. Sometimes, days are not mentioned explicitly: "In 2020, Alice earned $33200." Decomposing this sentence into events, we get

```
income_(span("earned", 15, 20)).
agent_(span("earned", 15, 20),
        span("Alice", 9, 13)).
amount_(span("earned", 15, 20),
         span(33200, 23, 27)).
start_(span("earned", 15, 20),
        span(D, S, E)).
```

By default, here, we set `D=20200101`. Any date that is the argument of a `start_` relation, and where only the year is specified, defaults to the first day of that year. Similarly, any date that is the argument of an `end_` relation and where only the year is specified, defaults to the last day of that year. This is enforced at annotation time. It is thus sufficient to find the string "2020" as the argument to the `start_` relation to get it right. `S` and `E` are set to the start and end of "2020": `S=3` and `E=6`.

B.2 Decoding

At decoding time, we look for arguments to `start_` and `end_` relations, and do our best to convert them to dates. The model of Section 4 returns spans. First, we get the surface form of that span, i.e. the string that was selected by the IE model. Next, we use the `datetime` package from Python and its `strptime` function to parse the string. We try an ordered list of possible regular expressions for dates, going from most specific ("August 24th, 1970") to least specific ("2020"). If one of them works, we will have a value at least for the year, and in the best case, values for year, month and day. From there, we can deduce missing values for month and day, following the convention with `start_` and `end_` mentioned above. If none of the regular expressions work, we look for a 4-digit integer as close as possible to the span, and use that as the year. Failing that, we set the year to 0. In our error analysis in Appendix E, we have found that dates were generally either extracted correctly, or not extracted at all, which suggests that the above heuristic is not a limiting factor in the performance.

C Probabilities of Knowledge Base Elements

As depicted in Figure 3, the tree can be represented as a set of pairs and triplets. Here, we will use the language of KBEs, which are either pairs (l, r) or triplets (l, r, c). Pairs (l, r) contain an event predicate l and a span r. Triplets (l, r, c) contain an argument predicate l, a span r, and a span c. Our model parameterizes their probability as follows.

For a pair, $p(l, r) = p(l|r)p(r)$. The probability of the span $p(r)$ is computed as the probability assigned to r by the BIO tagger $\mathcal{T}$ (line 6 of Algorithm 1). This is computed by summing the probabilities of all BIO sequences that allow r as a span. The probability $p(l|r)$ is assigned to event predicate l by the label classifier $\mathcal{L}$ (line 8). It is the probability that r is labeled with l.

For a triplet, $p(l, r, c) = p(l|r, c)p(c|r)p(r)$. The probability of the span $p(r)$ is given by the BIO tagger, as above. The probability of the second span $p(c|r)$ is given by the BIO tagger when conditioned on r. Here too, $p(c|r)$ is computed by summing the probabilities of all BIO sequences that allow c as a span. Finally, the probability of the argument predicate $p(l|r, c)$ is given by the classifier.

D Additional results

In Table 4, the target KB is the set of KBEs contained in the original SARA cases. We report precision, recall and F1 scores, for dev and test sets. Consequently,

- recall scores are systematically lower in Table 4 as compared to Table 2, and
- precision scores are identical.[8]

E Error analysis

We compare model outputs on 3 binary cases and 3 numerical cases picked at random from the test set. We only mention the errors made by the models. Table 5 shows the answers produced by the Prolog program and the KB on those 6 cases.

tax_case_78. This case has 6 events. LegalBERT misses one argument of the first out of two residence_ events, and overpredicts a second start_ date. LegalBERT mispredicts the patient_ of the second residence_ event, but nonetheless makes a sensible prediction. Together with LegalBERT', BERT and T5, LegalBERT correctly detects a payment_ event, even though that event had been discarded as part of pre-processing, as described in Section 5.1. For the first residence_ event, LegalBERT' mispredicts Bob as the agent_ instead of Charlie, predicts 2 start_ dates, and 2 patient_ arguments. For the second residence_ event, LegalBERT' predicts 2 agent_ and 2 patient_ arguments, one too many in each case. BERT, RoBERTa and T5 make similar predictions, getting all the event anchors right, but missing about half the arguments per event.

tax_case_34. This case contains a single income_ event. All 5 models perfectly predict the gold KB.

[8]Precision is higher in Table 4 for dev precision for LegalBERT. This difference comes from one out of the five runs, where AllenNLP returned precisions of 88.6 and 88.7. This is unexpected, and can presumably be ascribed to an artefact in rounding in Python. The remaining four runs have identical precision, and this single discrepancy led to different means, 88.74 and 88.76.

tax_case_28. This case contains 3 events. LegalBERT misses a token when predicting the amount_ argument of the payment_ event: it predicts “53” instead of “53200”. LegalBERT' perfectly predicts all 3 events and their arguments. BERT misses the purpose_ of service_, and predicts 2 amount_ arguments for the same income_ event, confusing amount_ of payment_ event and that of income_ event. RoBERTa predicts the income_ event and its arguments, but then additionally predicts 3 arguments: an extra agent_ (also Alice, but a different mention), start_ (some other date in the case), and amount_ (the amount_ of the income_ event). RoBERTa misses the purpose_ of the service_ event. Like BERT , RoBERTa predicts two amount_ arguments for the income_ event. T5 correctly predicts all 3 events, but misses half the arguments of the payment_ event, and misses the agent_ of the income_ event.

s68_a_2_pos. This case contains a single income_ event. All models perfectly predict the gold KB, except for T5 missing the agent_ argument, and BERT predicting that 2 different “Alice” spans are both agent_ arguments of the income_ event.

s3306_c_10_b_neg. This case contains 4 events. LegalBERT predicts 2 start_ dates to the medical_patient_ event. LegalBERT' fails to classify a hospital as a hospital, instead classifying it as an educational institution. It incorrectly infers start_ and end_ dates to the service_ event. BERT correctly classifies a hospital as a hospital, but misses an argument of that event. It fails to predict that Alice is a patient at this hospital, as well as what hospital someone is a patient at, and incorrectly infers start_ and end_ arguments to the payment_ event, like LegalBERT'. RoBERTa misses the agent_ of the hospital_ event, and predicts 2 start_ dates to medical_patient_. T5 correctly classifies the hospital, and otherwise misses or hallucinates one or two arguments per event. For example, it detects the fact that someone is a patient at a hospital, but fails to predict the agent_ of that relation, and predicts two start_ dates.

s152_a_neg. This case contains a single event: a brother_ relationship between Alice and Bob. LegalBERT, LegalBERT' and RoBERTa correctly predict the KB. BERT and T5 miss the patient_ of the relationship.

Model	Dev Precision	Recall	F1	Test Precision	Recall	F1
LegalBERT	**88.8** ± 0.6	**83.1** ± 0.6	**83.6** ± 0.3	**88.1** ± 0.2	**79.0** ± 0.3	**77.5** ± 0.3
LegalBERT'	87.4 ± 1.0	78.3 ± 1.3	80.2 ± 0.8	85.4 ± 0.9	75.4 ± 0.8	74.4 ± 0.6
BERT	85.9 ± 2.1	71.2 ± 2.4	75.0 ± 1.5	85.2 ± 2.3	69.4 ± 1.7	70.0 ± 1.4
RoBERTa	84.5 ± 0.6	76.2 ± 1.4	77.6 ± 0.8	86.4 ± 0.7	74.2 ± 1.1	73.4 ± 0.8
T5	73.2 ± 0.6	52.8 ± 2.1	58.8 ± 1.4	71.6 ± 1.1	52.8 ± 0.8	55.1 ± 0.8

Table 4: Performance metrics on the IE task measured on the full dataset. Scores are computed across 5 runs with different random seeds. We format statistics as *average* ± *standard error*. Best scores are **bolded**.

	`tax_case_78`	`tax_case_34`	`tax_case_28`	`s68_a_2_pos`	`s3306_c_10_B_neg`	`s152_a_neg`
LegalBERT	$15213	$2684	$344428	Yes	No	No
LegalBERT'	$15213	$2684	$344848	Yes	No	No
BERT	*$0*	$2684	$365916	Yes	No	No
RoBERTa	*$0*	$2684	$365916	Yes	No	No
T5	*$0*	$2684	*$0*	Yes	No	No
Empty	*$0*	$0	*$0*	Yes	No	No
Topline	$15213	$2684	$344848	Yes	No	No
Correct answer	$14470	$2684	$344848	Yes	No	No

Table 5: Answers obtained with the Prolog program and the KBs extracted with different encoders. The cases are those analyzed in Appendix E. Incorrect answers are in *red italics*.

F Hyperparameters

In Section 5.2, we use Optuna (Akiba et al., 2019) to find good hyperparameters for each pre-trained encoder, running 100 experiments per encoder. In Table 6, we report the range searched over for each hyperparameter. If there is a single value under "value range", it means this hyperparameter was set and not searched over with Optuna. Each experiment consists of 3 training runs with 3 different random seeds, using the average micro F1 score on the validation set.

We also report the hyperparameters of the best performing models, in Tables 7, 8, 9, 10, and 11. For each number, we report up to 5 significant digits.

G Data statistics

Figure 4 reports statistics on the dataset used for IE, as well as some examples.

Part of model	**Hyperparameter**	**Value range**
Encoder $\mathcal{E}$	Transformer model encoding the case description X	{`bert-base-cased`, `roberta-base`, `t5-base`, `zlucia/custom-legalbert`, `jhu-clsp/LegalBert`}
Distance embeddings $\mathcal{D}$	Size of the embeddings encoding the distance of each token to the event anchor	{8, 16, 32, 64,128, 256, 512}
Label classifier $\mathcal{L}$	Feature dropout in between the layers of the MLP	[0, 0.8]
	Number of units per layer	{32, 64, 128, 256, 512, 1024, 2048}
	Number of layers	{1, 2, 3, 4}
BIO tagger $\mathcal{T}$	Feature dropout in between the layers of the MLP	Always equal to the dropout of $\mathcal{L}$
	Number of units	{32, 64, 128, 256, 512, 1024, 2048}
	Number of layers	{1, 2, 3, 4}
Trainer	Batch size	{8, 16, 32, 64, 128, 256}
	Learning rate	[1e-6, 1e-3]
	Learning rate scheduler	`reduce_on_plateau` — this scheduler halves the learning rate when the validation score stops improving
	Optimizer	`huggingface_adamw`
	Maximum number of epochs to train for	200
	Patience	20
	Gradient clipping	1.0
Loss function	Loss trade-off — see λ in Section 4	[0, 1]

Table 6: Hyperparameter ranges in the hyperparameter search.

Part of model	**Hyperparameter**	**Value**
Encoder $\mathcal{E}$	Transformer model	`jhu-clsp/LegalBert`
Distance embeddings $\mathcal{D}$	Size	8
Label classifier $\mathcal{L}$	Feature dropout	$7.6067 \cdot 10^{-2}$
	Number of units	2048
	Number of layers	3
BIO tagger $\mathcal{T}$	Feature dropout	-
	Number of units	1024
	Number of layers	3
Trainer	Batch size	8
	Learning rate	$6.1891 \cdot 10^{-5}$
Loss function	Loss trade-off	$7.5288 \cdot 10^{-1}$

Table 7: Hyperparameters of best LEGALBERT model.

Part of model	**Hyperparameter**	**Value**
Encoder $\mathcal{E}$	Transformer model	`zlucia/custom-legalbert`
Distance embeddings $\mathcal{D}$	Size	8
Label classifier $\mathcal{L}$	Feature dropout	$4.7806 \cdot 10^{-1}$
	Number of units	2048
	Number of layers	1
BIO tagger $\mathcal{T}$	Feature dropout	-
	Number of units	2048
	Number of layers	3
Trainer	Batch size	8
	Learning rate	$3.8743 \cdot 10^{-5}$
Loss function	Loss trade-off	$9.3341 \cdot 10^{-1}$

Table 8: Hyperparameters of best LEGALBERT' model.

Part of model	**Hyperparameter**	**Value**
Encoder $\mathcal{E}$	Transformer model	`bert-base-cased`
Distance embeddings $\mathcal{D}$	Size	8
Label classifier $\mathcal{L}$	Feature dropout	$1.5865 \cdot 10^{-1}$
	Number of units	64
	Number of layers	1
BIO tagger $\mathcal{T}$	Feature dropout	-
	Number of units	256
	Number of layers	4
Trainer	Batch size	16
	Learning rate	$7.7282 \cdot 10^{-5}$
Loss function	Loss trade-off	$4.4946 \cdot 10^{-1}$

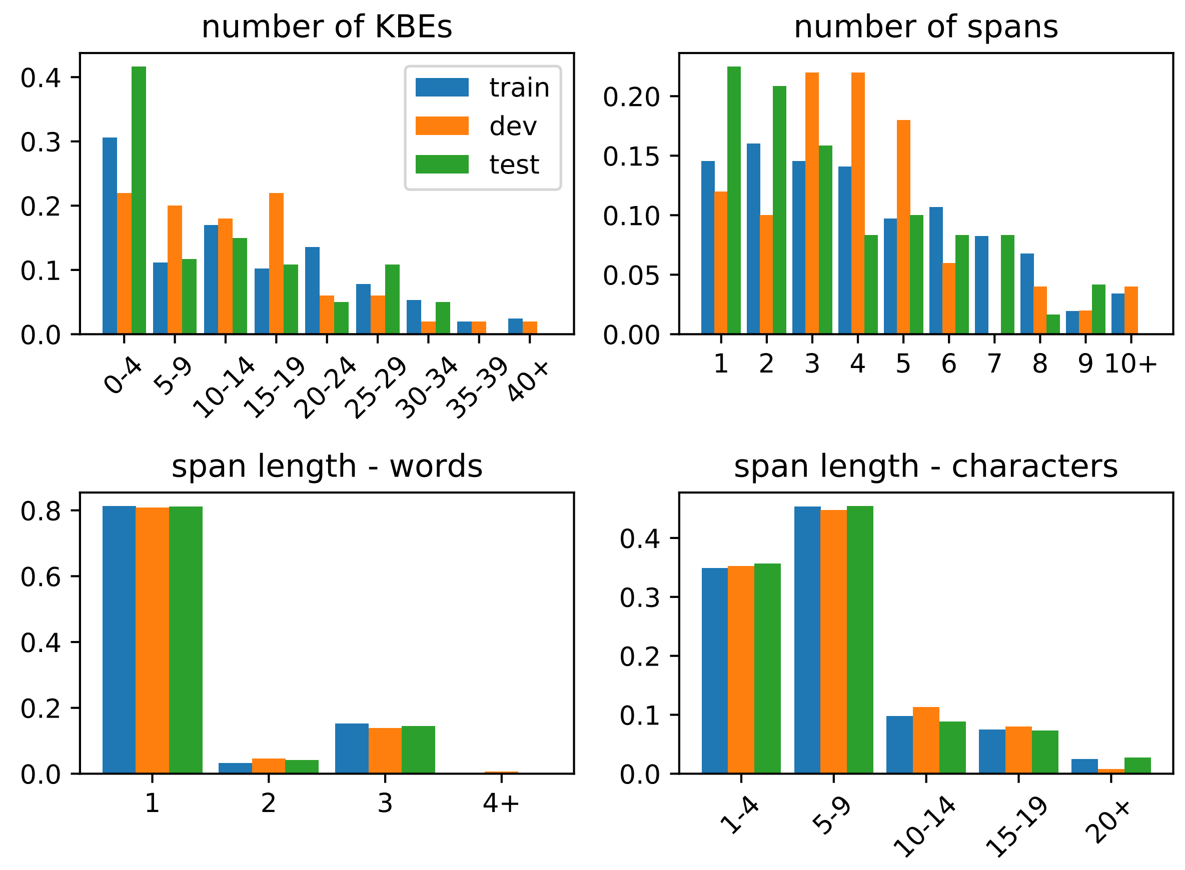
Table 9: Hyperparameters of best BERT model.

Part of model	**Hyperparameter**	**Value**
Encoder $\mathcal{E}$	Transformer model	`roberta-base`
Distance embeddings $\mathcal{D}$	Size	8
Label classifier $\mathcal{L}$	Feature dropout	$3.8621 \cdot 10^{-1}$
	Number of units	2048
	Number of layers	2
BIO tagger $\mathcal{T}$	Feature dropout	-
	Number of units	2048
	Number of layers	2
Trainer	Batch size	32
	Learning rate	$1.1132 \cdot 10^{-4}$
Loss function	Loss trade-off	$6.6690 \cdot 10^{-1}$

Table 10: Hyperparameters of best RoBERTa model.

Part of model	**Hyperparameter**	**Value**
Encoder $\mathcal{E}$	Transformer model	`t5-base`
Distance embeddings $\mathcal{D}$	Size	8
Label classifier $\mathcal{L}$	Feature dropout	$3.7523 \cdot 10^{-1}$
	Number of units	64
	Number of layers	1
BIO tagger $\mathcal{T}$	Feature dropout	-
	Number of units	1024
	Number of layers	1
Trainer	Batch size	32
	Learning rate	$5.6507 \cdot 10^{-4}$
Loss function	Loss trade-off	$5.0650 \cdot 10^{-1}$

Table 11: Hyperparameters of best T5 model.



Examples of spans of length 1, 2 and 3: “Alice”, “Bob”, “marriage”, “joint return”, “Feb 3rd, 1992” and “decree of divorce”.

Figure 4: Statistics of the data used for IE (partial dataset). Number of KBEs and spans are per case. The large number of 3-word spans is due to many date expressions being 3-word phrases.