# Named Entity Recognition in Indian court judgments

**Prathamesh Kalamkar**[1,2,*], **Astha Agarwal**[1,2,*], **Aman Tiwari**[1,2,*], **Smita Gupta**[3,*], **Saurabh Karn**[3,*], **Vivek Raghavan**[1]
[1]EkStep Foundation, [2]Thoughtworks Technologies India Pvt Ltd., [3]Agami
{prathamk, aman.tiwari, astha.agarwal}@thoughtworks.com,
{smita, saurabh}@agami.in, vivek@ekstep.org

## Abstract

Identification of named entities from legal texts is an essential building block for developing other legal Artificial Intelligence applications. Named Entities in legal texts are slightly different and more fine-grained than commonly used named entities like Person, Organization, Location etc. In this paper, we introduce a new corpus of 46545 annotated legal named entities mapped to 14 legal entity types. The Baseline model for extracting legal named entities from judgment text is also developed. We publish the training, dev data and trained baseline model https://github.com/Legal-NLP-EkStep/legal_NER.

## 1 Introduction

Artificial Intelligence has the potential to increase access to justice and make various legal processes more efficient (Zhong et al., 2020). Populous countries such as India have a problem with high case pendency. As of March 2022, over 47 million cases are pending in Indian courts[1]. Hence, it becomes imperative to use AI to reduce the strain on the judicial system and reduce pendency. For developing legal AI applications, it is essential to have access to judicial data and open-source foundational AI building blocks like Named Entity Recognition (NER). A lot of Indian legal data is publicly available thanks to open data initiatives like National Judicial Data Grid (NJDG) and the Crime and Criminal Tracking Network and System (CCTNS).

NJDG provides non-exhaustive metadata of Indian court judgements like the names of petitioners, respondents, lawyers, judges, date, court etc. Extracting these entities from judgment text makes the information extraction exhaustive and reduces errors like misspellings compared to NJDG metadata. Helpful information like precedents and statutes are also not written in the NJDG metadata. Hence it is essential to extract from court judgment texts rather than just relying on the published NJDG metadata. Extracting named entities from the text also paves the foundation for more tasks like relation extraction, coreference resolution, knowledge graph creation etc.

In this paper, we have created a corpus of annotated judgment texts with 14 legal entities (details in §3). An example of annotated entities is shown in Figure 1.

We make the following contributions in this paper

- We create a corpus of 14444 Indian court judgment sentences and 2126 judgment preambles annotated with 14 legal named entities.
- We develop a transformer-based legal NER baseline model
- We create rule-based post-processing, which captures the document level context and coreference resolution for certain entities
- A representative sample of Indian high court and supreme court judgments having 11970 judgments across 29 Indian courts

## 2 Related Work

Named Entity Recognition (NER) is widely studied in literature ranging from statistical models (Borthwick et al., 1998),(Bikel et al., 1999),(McCallum and Li, 2003) to state-of-the-art deep neural nets (Li et al., 2020). The task complexity is also evolved over time from flat named entities to nested entities, from monolingual to multilingual NER.

Legal domain-specific entities are often used for more meaningful information extraction from legal texts. Pioneering work in legal NER by (Dozier et al., 2010) developed named entity recognition & resolution system on US legal texts using 5 legal named entities (judges, attorneys, companies,

* Authors contributed equally
[1]https://www.livelaw.in/pdf_upload/au595-426886.pdf



*Proceedings of the Natural Legal Language Processing Workshop 2022*, pages 184 - 193

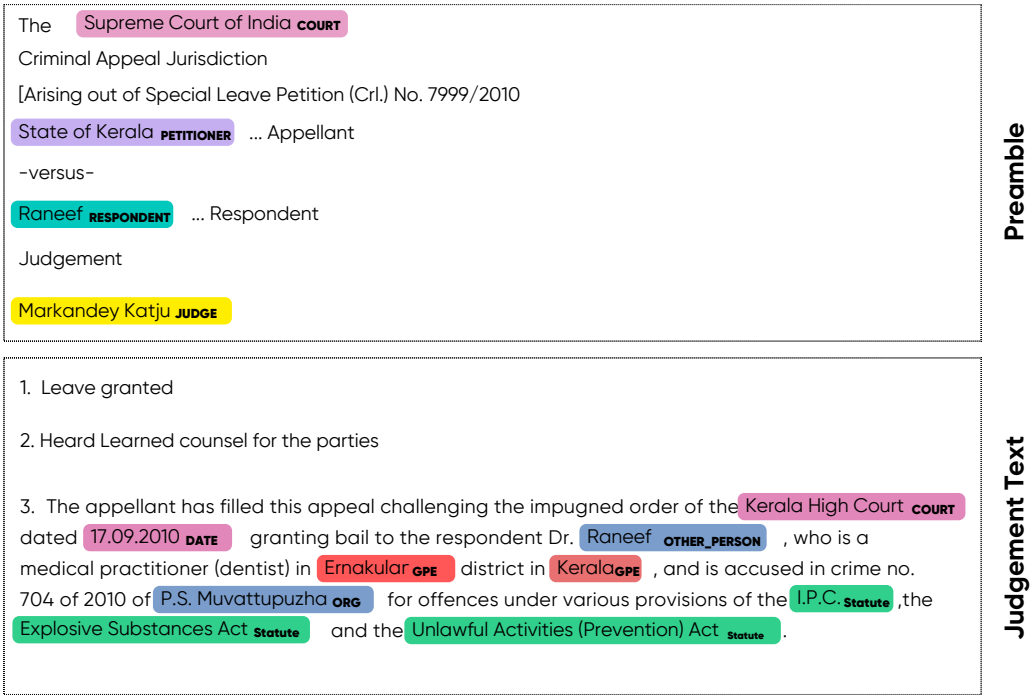


Figure 1: Legal Named Entities in a court judgment

jurisdictions, and courts). (Cardellino et al., 2017) created Named Entity Recognizer, Classifier and Linker by mapping LKIF ontology to YAGO ontology using Wikipedia data and various levels of abstraction of the legal ontology. Since the legal vocabulary and style of writing of legal text varies by language and geography, it is often necessary to create separate datasets and models. (Glaser et al., 2018) compared GermaNER (Benikova et al., 2015) and DBpedia Spotlight (Mendes et al., 2011; Daiber et al., 2013) NER systems on German legal contracts. (Leitner et al., 2020) created a German NER dataset with 19 fine-grained semantic classes. (Păiș et al., 2021) created a Romanian legal corpus called Legal NERo, which has 370 documents annotated with five entity classes and used legal domain word embeddings to build the NER system. (Luz de Araujo et al., 2018) created a corpus of legal documents from several Brazilian Courts called LeNER-Br, which is annotated with six entity classes. (Angelidis et al., 2018) created Named Entity Recognizer and Linker for Greek legislation with 254 annotated pieces of legislation. (Chalkidis et al., 2021) extracted contract elements extraction using LSTM encoders. NER using contextual dictionaries was applied to the French legal corpus of 94 court judgments with four entity classes by (Barriere and Fouret, 2019). As a part of Lynx, project (Schneider et al., 2020), a set of services, including NER, were developed to help create a legal domain knowledge graph and its use for the semantic analysis of legal documents.

Transition-based parsing for NER was proposed by (Lample et al., 2016) using stacked LSTM. NER task can be treated as graph-based dependency parsing (Yu et al., 2020) to provide a global view of the input using biaffine model. Recent advances in span representation have shown promising results for Named Entity Recognition (Ouchi et al., 2020). Span pretraining methods (Joshi et al., 2020) improve the span representation for pre-trained language models via span-level pretraining tasks. Infusing external knowledge for entity representation and linking (Yamada et al., 2020), (Wang et al., 2021) helps to better represent the knowledge in legal texts. (Ye et al., 2022) considered interrelation between spans by considering the neighbouring spans integrally to better model the entity boundary information.

Recently a lot of work has been done in the legal AI field in the Indian context. Structuring the court judgments (Kalamkar et al., 2022), legal statute

identification (Paul et al., 2022a), judgment outcome prediction (Malik et al., 2021), judgment summarization (Shukla et al., 2022) provide AI building blocks. (Paul et al., 2022b) created InLegalBERT and InCaseLawBERT, which are further pre-trained versions of LegalBERT (Chalkidis et al., 2020) and CaseLawBERT (Zheng et al., 2021) respectively on Indian legal text.

## 3 Legal Named Entity Recognition Corpus

### 3.1 Legal Named Entities

A typical Indian court judgment can be split into two parts viz., preamble and judgment. The preamble of a judgment contains formatted metadata like names of parties, judges, lawyers, date, court etc. The text following the preamble till the end of the judgment is called "judgment". An example showing the preamble and judgment of a court judgment along with entities is shown in Figure 1. The preamble typically ends with keywords like JUDGMENT or ORDER etc. In case these keywords are not found, we treat the first occurrence of 2 consecutive sentences with a verb as the start of the judgment part. This is because the preamble typically contains formatted metadata and not grammatically complete sentences.

After discussion with legal experts about the useful information to be extracted from court judgments, we came up with a list of legal named entities which are described in Table 1. Some entities are extracted from the preamble, and some from the judgment text. Some entities are extracted from both the preamble and judgment, and their definitions may change depending on where they are extracted from.

Flat entities were considered for annotation i.e., "Bank of China" should be considered as an ORG entity and "China" should not be marked as GPE inside this entity. The detailed definitions with correctly and incorrectly marked examples can be found here[2].

### 3.2 Representative Sample of Indian High Court & Supreme Court judgments

Selecting a representative sample of court judgments text is vital to cover varieties of styles of writing judgments. Most cited judgements are likely to be more important for applying the NER model. But just taking the most cited judgments from a given court would produce bias in certain types of cases. Hence it is necessary to control case types as well. We created the following 8 types of cases (Tax, Criminal, Civil, Motor Vehicles, Land & Property, Industrial & Labour, Constitution and Financial) which cover most of the cases in Indian courts. Classification of each judgment into one of these 8 types is a complex task. We have used a naive approach to use keywords based on act names for assigning a judgment to a case type. E.g., If the judgment mentions the "income tax act" then most probably it belongs to the "Tax" category. We use IndianKanoon search engine[3] to get the most cited court judgments matching the key act names. The key act names for each of the case types are given in Table 2.

One IndianKanoon search query was created for each of the 8 case types and 29 courts (supreme court, 23 high courts, three tribunals and 2 district courts). The Topmost cited results from each query were combined and de-duplicated to produce the final corpus of judgments. We consider judgments in the English language only. Judgments obtained by this method from 1950 to 2017 were used for training data annotations, and judgments from 2018 to March 2022 were used for the test and dev data annotations. The representative sample dataset of 11970 judgments, along with search queries, the full text of the judgments and descriptive statistics, are published in our git repository[4]. We believe these representative judgments can be used for other future studies as well.

### 3.3 Data Annotation Process

The annotations for judgment text were done at a sentence level, i.e. separate individual judgment sentences were presented for annotation without the document-level context. However, annotators had the freedom to access the entire judgment text by clicking on the Indiankanoon URL shown below the text in case they needed more context. Complete preambles were presented for annotation.

#### 3.3.1 Selecting Raw Text to Annotate

Legal named entities in a judgment text tend to be sparse, i.e., many of the sentences in a court judgment may not have any legal named entities. Hence is essential to identify entity-rich sen-

[2] https://storage.googleapis.com/indianlegalbert/OPEN_SOURCED_FILES/NER/NER_Definitions.pdf

[3] https://indiankanoon.org/

[4] https://github.com/Legal-NLP-EkStep/legal_NER/tree/main/representative_judgments_sample

| Named Entity | Extract From | Description |
|---|---|---|
| COURT | Preamble, Judgment | Name of the court which has delivered the current judgement if extracted from the preamble. Name of any court mentioned if extracted from judgment sentences. |
| PETITIONER | Preamble, Judgment | Name of the petitioners/appellants/revisionist from current case |
| RESPONDENT | Preamble, Judgment | Name of the respondents/defendants/opposition from current case |
| JUDGE | Preamble, Judgment | Name of the judges from the current case if extracted from the preamble. Name of the judges of the current as well as previous cases if extracted from judgment sentences. |
| LAWYER | Preamble | Name of the lawyers from both the parties |
| DATE | Judgment | Any date mentioned in the judgment |
| ORG | Judgment | Name of organizations mentioned in text apart from the court. |
| GPE | Judgment | Geopolitical locations which include names of states, cities, villages |
| STATUTE | Judgment | Name of the act or law mentioned in the judgement |
| PROVISION | Judgment | Sections, sub-sections, articles, orders, rules under a statute |
| PRECEDENT | Judgment | All the past court cases referred to in the judgement as precedent. Precedent consists of party names + citation(optional) or case number (optional) |
| CASE_NUMBER | Judgment | All the other case numbers mentioned in the judgment (apart from precedent) where party names and citation is not provided |
| WITNESS | Judgment | Name of witnesses in current judgment |
| OTHER_PERSON | Judgment | Name of all the persons that are not included in petitioner, respondent, judge and witness |

Table 1: Legal Named Entities Definitions

| Case Type | Key Act keywords |
|---|---|
| Tax | tax act, excise act, customs act, goods and services act etc. |
| Criminal | IPC, penal code, criminal procedure etc. |
| Civil | civil procedure, family courts, marriage act, wakf act etc. |
| Motor Vehicles | motor vehicles act, mv act, imv act etc. |
| Land & Property | land acquisition act, succession act, rent control act etc. |
| Industrial & Labour | companies act, industrial disputes act, compensation act etc. |
| Constitution | constitution |
| Financial | negotiable instruments act, sarfaesi act, foreign exchange regulation act etc. |

Table 2: Key Act Names for Each Case Type

tences for annotation rather than taking a random sample. We used the spacy pre-trained model (en_core_web_trf) (Montani et al., 2022) with custom rules to predict the legal named entities. Custom rules were used to map the Spacy-defined named entities to the legal named entities defined in this paper. E.g., An entity predicted by spacy as PERSON with the keyword "petitioner" nearby was marked as PETITIONER etc. We passed the representative sample judgment texts through this Spacy model with custom rules to get predicted noisy legal entities. Using these predicted legal entities, we selected the sentences that are entity-rich and that reduce the class imbalance across different entity types. We also added sentences without any predicted entities. Very short sentences and sentences with non-English characters were discarded. Preambles, where party names are written side by side on the same line, were also discarded.

### 3.3.2 Pre-annotations

The data annotation was done in 4 cycles. The preambles and sentences were pre-annotated in each cycle to reduce annotation effort.

For the first annotation cycle, the predicted legal entities obtained during the raw text selection process, as mentioned in 3.3.1, were reviewed and corrected. At the end of cycle 1, a machine learning model using Roberta+ transition-based parser architecture (explained in detail in §4) was trained using the labelled data obtained in cycle 1. This machine learning model was used to pre-annotate the cycle 2 data. Similarly, the machine learning model trained using cycle 1 and 2 data was used to pre-annotate cycle 3 data and so on.

### 3.3.3 Manual Reviews & Corrections

In each cycle, all of the pre-annotated preambles and sentences were carefully reviewed and corrected by humans. Roughly the same amount of preamble and sentences were annotated in each cycle. The team of 4 legal experts and 4 data scientists at OpenNyAI did the data annotation. Legal experts were law students from various law universities across India. We did not do duplicate annotations to maximize the number of annotated data. We used the Prodigy tool[5] for the annotations.

The corrected data obtained from the four annotation cycles was split into the train, dev and test datasets as per the time ranges mentioned in Table 3. We tried to keep the dev data distribution similar to the test data distribution. Test and dev data was carefully cross-reviewed twice to ensure data quality. The total count of entities, total number of preambles and judgment sentences in train, dev and test data are shown in Table 3. The counts of each legal named entity in training data are shown in Table 4.

| | **Train** | **Dev** | **Test** |
|---|---|---|---|
| Time Range | 1950 to 2017 | 2018 to 2022 | 2018 to 2022 |
| Preambles | 1560 | 125 | 441 |
| Judgment sentences | 9435 | 949 | 4060 |
| Entities | 29964 | 3216 | 13365 |

Table 3: Train & Test data counts

[5]https://prodi.gy/

| **Entity** | **Judgment Count** | **Preamble Count** |
|---|---|---|
| COURT | 1293 | 1074 |
| PETITIONER | 464 | 2604 |
| RESPONDENT | 324 | 3538 |
| JUDGE | 567 | 1758 |
| LAWYER | NA | 3505 |
| DATE | 1885 | NA |
| ORG | 1441 | NA |
| GPE | 1398 | NA |
| STATUTE | 1804 | NA |
| PROVISION | 2384 | NA |
| PRECEDENT | 1351 | NA |
| CASE_NUMBER | 1040 | NA |
| WITNESS | 881 | NA |
| OTHER_PERSON | 2653 | NA |
| **Total** | 17485 | 12479 |

Table 4: Counts of Legal Entities for Training data in Preamble & Judgment

## 4 NER Baseline Model

The end goal behind this work is to enable the development of other legal AI applications that consume automatically detected legal named entities from judgment texts. Towards this goal, we experimented with some famous NER model architectures. A single model was trained to predict entities from both judgment sentences and the preamble. As transformer-based architectures have shown a lot of success in NER tasks (Li et al., 2020), we mainly experimented with them. We compared the performance of 2 NER architecture types when trained on our legal NER dataset. The first architecture type uses a transition-based dependency parser (Honnibal and Johnson, 2015) on top of the transformer model. The second architecture type uses a fine-tuning based approach which adds a single linear layer to the transformer model and fine-tunes the entire architecture on the NER task. Figure 2 shows the 2 NER architecture types.

We experimented with multiple transformer models for each of the architecture types. For transition-based parser architecture we experimented with the Roberta-base model (Liu et al., 2019), InLegalBERT (Paul et al., 2022b) using Spacy library. For the fine-tuning approach we experimented with Roberta-base, InLegalBERT, legalBERT (Chalkidis et al., 2020) using TNER library (Ushio and Camacho-Collados, 2021).

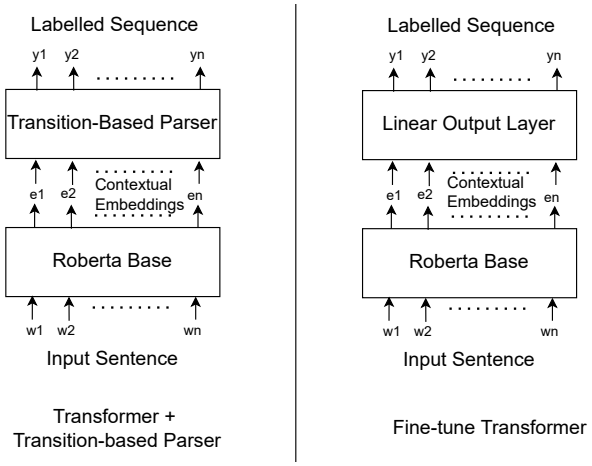


Figure 2: NER Architectures

| **Architecture Type** | **Trans. Model** | **P** | **R** | **F1** |
|---|---|---|---|---|
| Transformer + Transition Based Parser | **Roberta-base** | **92.0** | **90.2** | **91.1** |
| | InLegal BERT | 87.3 | 85.8 | 86.5 |
| Fine Tune Transformer | Roberta-base | 77.6 | 80.0 | 78.8 |
| | InLegal BERT | 77.7 | 84.6 | 81.0 |
| | Legal BERT | 75.4 | 79.5 | 77.5 |

Table 5: Model Performance on test data

The models are evaluated by using recall, precision and strict F1 scores on combined preamble and judgment sentences. The named entity is considered correct when both boundary and entity class are predicted correctly. Table 5 shows the performance of these experiments on the test data.

Performance of the best performing model (Roberta+ transition-based parser) on each of the entity classes on test data along with average character length is shown in Table 6. It also shows the Type match F1 score which was proposed in (Segura-Bedmar et al., 2013). Under the Type match evaluation scheme, some overlap between the tagged entity and the gold entity is required along with entity type match. In strict f1 calculation, the entities with correct entity type match and partial span overlap are considered incorrect. But in the Type match evaluation, such entities are considered the correct entity. Hence the Type match F1 score gives an indication of how much overlap exists between ground truth and prediction considering partial matches.

| **Entity** | **Count** | **Avg. Len.** | **F1** | **Type match F1** |
|---|---|---|---|---|
| COURT | 1231 | 25 | 95.4 | 97.2 |
| PETITIONER | 835 | 20 | 89.8 | 92.6 |
| RESPONDENT | 1125 | 34 | 83.0 | 91.8 |
| JUDGE | 580 | 15 | 95.4 | 96.5 |
| LAWYER | 1813 | 16 | 94.1 | 95.5 |
| DATE | 1111 | 11 | 91.9 | 98.7 |
| ORG | 920 | 18 | 86.4 | 90.2 |
| GPE | 711 | 8 | 85.7 | 90.9 |
| STATUTE | 971 | 17 | 96.0 | 97.6 |
| PROVISION | 1220 | 14 | 95.7 | 98.6 |
| PRECEDENT | 634 | 62 | 80.1 | 96.2 |
| CASE_NUMBER | 683 | 23 | 89.1 | 92.4 |
| WITNESS | 446 | 12 | 89.7 | 89.7 |
| OTHER_PERSON | 1085 | 12 | 93.8 | 95.2 |
| **Overall** | **13365** | **20** | **91.1** | **94.9** |

Table 6: Entity-wise performance of Roberta + Transition-based Parser model on test data

| **Parameter** | **Value** |
|---|---|
| Transformer | Roberta-base |
| Optimizer | Adam with beta1 = 0.9, beta2 = 0.999, L2 = 0.01, initial learning rate = 0.00005 |
| max training steps | 40000 |
| training batch size | 256 |

Table 7: Key training procedure parameters

The trained Roberta+Transition-based Parser is made available as a Spacy pipeline in our git repository and hugging face model repository[6].

### 4.1 Training Procedure

Early stopping using dev data was used during training to select the best epoch for all the experiments. The details about the training procedure for Roberta + Transition-based parser using Spacy are available in the GitHub repository. NVIDIA Tesla V100 GPU was used to train the model and the training time was 12 hours. The key parameters used are mentioned in Table 7.

[6]https://huggingface.co/opennyaiorg/en_legal_ner_trf

### 4.2 Results Discussion

Adding a transition-based parser to the transformer architecture significantly improves the model's accuracy for this NER task, as seen in Table 5.

As seen from Table 6, the Roberta-base + transition-based parser NER model can extract shorter entities like WITNESS, PROVISION, STATUTE, LAWYER, COURT and JUDGE with excellent performance.

PRECEDENT has degraded performance as compared to other entities because the precedent names are usually very long (average entity length is 62 characters) and missing out on even a few characters makes the entire entity to be marked as incorrect. Because of this reason, there is a significant difference between strict F1 and Type match F1 for PRECEDENT. Manual inspection of errors in PRECEDENT prediction reveals that many a time, the prefixes of party names like "Mr.", "M/S", etc. are missed in the prediction while gold entities have them. E.g., the gold entity type is PRECEDENT with the text "Mr Amit Kumar Vs State of Maharashtra," and the predicted entity type is PRECEDENT with the text "Amit Kumar Vs State of Maharashtra". In strict F1 evaluation, this example is considered incorrect, while in Type match evaluation, this example is considered correct. One possible reason for the model not to include the prefixes in the PRECEDENT prediction could be that prefixes are not considered as a part of other entities like PETITIONER, RESPONDENT and ORG. So possibly, the model has also learned to omit the prefixes in PRECEDENT.

The average character length of RESPONDENT entities is considerably higher than that of PETITIONER entities. This difference is because, often, the respondents are posts or authorities rather than a person. E.g. "The Chief Engineer, Water Resource Organization, Chepauk, Chennai-5". In such cases, gold data marks the authority or post names along with the address as the corresponding entity, making them longer. The difference between strict F1 and Type match F1 for RESPONDENT shows that the model is missing in predicting a few characters in such long entities.

The overall accuracy of this model makes it very useful in practical legal AI applications.

## 5 Post-Processing of Named Entities

Since the annotators were asked to annotate individual sentences without document-level context, any trained NER model on this data will also focus only on the sentence-level information. While inferring the NER model on a complete judgment text, it is important to perform post-processing of extracted entities to capture document-level context. In particular, we create rules to perform the following tasks

- Reconciliation of the entities extracted from individual sentences of a judgment
- Coreference resolution of precedents
- Coreference resolution of statutes
- Assign statute to every provision

### 5.1 Reconciliation of the Extracted Entities

The same entity text can be tagged with different legal entity classes in separate sentences of the same judgment. E.g., In the preamble of a judgment, it is written that "Amit Kumar" is a petitioner. In the same judgement text, the judge later writes, "Four unidentified persons attacked Amit Kumar". NER model would mark "Amit Kumar" in the second mention as OTHER_PERSON because there is no information about Amit Kumar being a petitioner in this sentence. Marking this person's name as PETITIONER is more valuable than marking it as an OTHER_PERSON.

As part of entity reconciliation, entities predicted as OTHER_PERSON or ORG are matched with all the PETITIONER, RESPONDENT, JUDGE, LAWYER and WITNESS entities. If an exact match is found, then the entity type is overwritten with the matching entity type. In the previous example, all the extracted entities that match "Amit Kumar" would be overwritten with entity type PETITIONER.

### 5.2 Coreference Resolution of Precedents

The names of precedent cases are usually very long. Hence judges typically mention the complete name of a precedent case for the first mention and later use the name of the first party as a reference. E.g., "The constitution bench of this court in Gurbaksh Singh Sibbia and others Vs State of Punjab (1980) 2 SCC 565 dealt with the scope and ambit of anticipatory bail". Then, later on, the judge uses a reference to this case, like "The learned counsel for the petitioner placed reliance on Sibbia's case (supra)." The NER model identifies "Sibbia" as OTHER_PERSON in the second sentence. But

here, "Sibbia" is a reference to the earlier extracted PRECEDENT entity "Gurbaksh Singh Sibbia and others Vs State of Punjab (1980) 2 SCC 565".

We first cluster all the extracted precedent entities within a judgment by matching the party names and citations. A precedent cluster contains all the precedent entities with matching party names or citations. Then we identify potential precedent referents as ORG or OTHER_PERSON entity followed by keywords "supra" or "'s case". We then search such referent entities in the extracted precedents' party names and find the closest matching preceding precedent. If the match is found, then we change the referent entity type to PRECEDENT. Referent entities are also added to the precedent cluster where the closest matching precedent belongs. Once all the matching precedent referents are assigned to precedent clusters, the longest entity in each cluster is marked as the cluster head. So in the example before, the entity type for "Sibbia" in the second sentence would be changed from OTHER_PERSON to PRECEDENT, and a precedent cluster would be created with the head as "Gurbaksh Singh Sibbia and others Vs State of Punjab (1980) 2 SCC 565" and the member as "Sibbia".

The information about precedents coreference can be accessed through output Spacy doc object property *doc.user_data['precedent_clusters']*.

### 5.3 Coreference Resolution of Statutes

Statute names can be long and are frequently mentioned in judgment text. Hence judges typically write the complete statute name at the beginning of the judgment and specify the referent for this statute for the remaining judgement. E.g., "The complaint was filed under the Companies Act, 1956 (for brevity, 'the Act') ...". Later on in the same judgment, the judge writes ", Section 5 of the Act defines ...". We write rules to identify such statute referents by searching for a STATUTE entity followed by keywords in parenthesis. Such statute referents are added to the statute cluster with its head as the complete statute name. All entities in a statute cluster refer to the same statute, which is the head of the cluster. Extracted statutes are also looked up against a list of famous acronyms (IPC, CrPC etc.), and if a match is found, then the corresponding full form is added to the statutes cluster. The information about statute coreference can be accessed through output Spacy doc object property *doc.user_data['statute_clusters']*.

### 5.4 Assign Statute to Every Provision

Every extracted provision should be associated with an extracted statute. Sometimes a provision and its corresponding statute are explicitly mentioned in the same sentence. E.g., "Section 420 of Indian Penal Code says ...". Sometimes, the provision-statute mapping is implicit where only the provision is mentioned, and the corresponding statute is understood from the context. E.g., "The section 420 says ...".

In case of explicit mentions, we assign the statute to the immediately preceding provision in the same sentence. All the remaining provisions are considered implicit provisions. We first search for all the implicit provisions if a unique explicit mapping exists in another sentence. E.g., if the judge writes an explicit mention like "Section 420 of Indian Penal Code" and there is no other explicit mention of Section 420 for any other statute in the entire judgment text, then all the implicit mentions of Section 420 are mapped to "Indian Penal Code". Suppose no explicit mention for a provision is found, or multiple explicit mentions are found for a provision. In that case, the statute extracted from the closest preceding sentence is assigned.

The assignment of the statute to provisions can be accessed via output Spacy doc object property *doc.user_data['provision_statute_pairs']*

## 6 Conclusion & Future Directions

In this paper, we proposed a new corpus of legal named entities using 14 legal entity types. We also proposed baseline models trained using this corpus along with post-processing of the extracted entities to capture document-level information. We have also released a representative sample of Indian court judgments which could be used in further studies. We believe this corpus will lay the foundation for further NLP tasks like relationship extraction, knowledge graph population etc. using Indian court judgments.

## Acknowledgements

This work is part of the OpenNyAI mission, which is funded by EkStep and Agami. We thank all the law experts, student volunteers for contributing to data annotation.

## References


Iosif Angelidis, Ilias Chalkidis, and Manolis Koubarakis. 2018. Named entity recognition, linking and generation for greek legislation. In *JURIX*, pages 1–10.

Valentin Barriere and Amaury Fouret. 2019. May i check again?—a simple but efficient way to generate and use contextual dictionaries for named entity recognition. application to french legal texts. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 327–332.

Darina Benikova, Seid Muhie, Yimam Prabhakaran, and Santhanam Chris Biemann. 2015. C.: Germaner: Free open german named entity recognition tool. In *In: Proc. GSCL-2015*. Citeseer.

Daniel M Bikel, Richard Schwartz, and Ralph M Weischedel. 1999. An algorithm that learns what's in a name. *Machine learning*, 34(1):211–231.

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. 1998. Nyu: Description of the mene named entity system as used in muc-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*.

Cristian Cardellino, Milagro Teruel, Laura Alonso Alemany, and Serena Villata. 2017. A low-cost, high-coverage legal named entity recognizer, classifier and linker. In *Proceedings of the 16th edition of the International Conference on Articial Intelligence and Law*, pages 9–18.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2021. Neural contract element extraction revisited: Letters from sesame street. *arXiv preprint arXiv:2101.04355*.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th international conference on semantic systems*, pages 121–124.

Christopher Dozier, Ravikumar Kondadadi, Marc Light, Arun Vachher, Sriharsha Veeramachaneni, and Ramdev Wudali. 2010. Named entity recognition and resolution in legal text. In *Semantic Processing of Legal Texts*, pages 27–43. Springer.

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.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1373–1378.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Prathamesh Kalamkar, Aman Tiwari, Astha Agarwal, Saurabh Karn, Smita Gupta, Vivek Raghavan, and Ashutosh Modi. 2022. Corpus for automatic structuring of legal documents. In *Proceedings of the Language Resources and Evaluation Conference*, pages 4420–4429, Marseille, France. European Language Resources Association.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Elena Leitner, Georg Rehm, and Julian Moreno Schneider. 2020. A dataset of german legal documents for named entity recognition. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4478–4485.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70.

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. *arXiv preprint arXiv:1907.11692*.

Pedro Henrique Luz de Araujo, Teófilo E de Campos, Renato RR de Oliveira, Matheus Stauffer, Samuel Couto, and Paulo Bermejo. 2018. Lener-br: a dataset for named entity recognition in brazilian legal text. In *International Conference on Computational Processing of the Portuguese Language*, pages 313–323. Springer.

Vijit Malik, Rishabh Sanjay, Shubham Kumar Nigam, Kripabandhu Ghosh, Shouvik Kumar Guha, Arnab Bhattacharya, and Ashutosh Modi. 2021. Ildc for cjpe: Indian legal documents corpus for court judgment prediction and explanation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4046–4062.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random

fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191.

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8.

Ines Montani, Matthew Honnibal, Matthew Honnibal, Sofie Van Landeghem, Adriane Boyd, Henning Peters, Paul O'leary McCann, Maxim Samsonov, Jim Geovedi, Jim O'Regan, Duygu Altinok, György Orosz, Søren Lind Kristiansen, Roman, Explosion Bot, Lj Miranda, Leander Fiedler, Daniël de Kok, Grégory Howard, Edward, Wannaphong Phatthiyaphaibun, Yohei Tamura, Sam Bozek, murat, Mark Amery, Ryn Daniels, Björn Böing, Pradeep Kumar Tippa, and Peter Baumgartner. 2022. explosion/spacy: v3.2.4: Workaround for Click/Typer issues.

Hiroki Ouchi, Jun Suzuki, Sosuke Kobayashi, Sho Yokoi, Tatsuki Kuribayashi, Ryuto Konno, and Kentaro Inui. 2020. Instance-based learning of span representations: A case study through named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6452–6459.

Vasile Păiș, Maria Mitrofan, Carol Luca Gasan, Vlad Coneschi, and Alexandru Ianov. 2021. Named entity recognition in the romanian legal domain. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 9–18.

Shounak Paul, Pawan Goyal, and Saptarshi Ghosh. 2022a. Lesicin: A heterogeneous graph-based approach for automatic legal statute identification from indian legal documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11139–11146.

Shounak Paul, Arpan Mandal, Pawan Goyal, and Saptarshi Ghosh. 2022b. Pre-training transformers on indian legal text.

Julian Moreno Schneider, Georg Rehm, Elena Montiel-Ponsoda, Víctor Rodríguez Doncel, Artem Revenko, Sotirios Karampatakis, Maria Khvalchik, Christian Sageder, Jorge Gracia, and Filippo Maganza. 2020. Orchestrating nlp services for the legal domain. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2332–2340.

Isabel Segura-Bedmar, Paloma Martínez, and María Herrero-Zazo. 2013. SemEval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (DDIExtraction 2013). In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 341–350, Atlanta, Georgia, USA. Association for Computational Linguistics.

Abhay Shukla, Paheli Bhattacharya, Soham Poddar, Rajdeep Mukherjee, Kripabandhu Ghosh, Pawan Goyal, and Saptarshi Ghosh. 2022. Legal case document summarization: Extractive and abstractive methods and their evaluation. In *The 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*.

Asahi Ushio and Jose Camacho-Collados. 2021. T-NER: An all-round python library for transformer-based named entity recognition. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 53–62, Online. Association for Computational Linguistics.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454.

Deming Ye, Yankai Lin, Peng Li, and Maosong Sun. 2022. Packed levitated marker for entity and relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4904–4917.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476.

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 *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pages 159–168.

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*, pages 5218–5230.