LeSICiN: A Heterogeneous Graph-based Approach for Automatic Legal Statute Identification from Indian Legal Documents

Shounak Paul,[1] **Pawan Goyal,** [1] **Saptarshi Ghosh** [1]
[1] Indian Institute of Technology, Kharagpur
shounakpaul95@kgpian.iitkgp.ac.in, pawang@cse.iitkgp.ac.in, saptarshi@cse.iitkgp.ac.in

Abstract

The task of Legal Statute Identification (LSI) aims to identify the legal statutes that are relevant to a given description of facts or evidence of a legal case. Existing methods only utilize the textual content of facts and legal articles to guide such a task. However, the citation network among case documents and legal statutes is a rich source of additional information, which is not considered by existing models. In this work, we take the first step towards utilising both the text and the legal citation network for the LSI task. We curate a large novel dataset for this task, including facts of cases from several major Indian Courts of Law, and statutes from the Indian Penal Code (IPC). Modeling the statutes and training documents as a heterogeneous graph, our proposed model **LeSICiN** can learn rich textual and graphical features, and can also tune itself to correlate these features. Thereafter, the model can be used to inductively predict links between test documents (new nodes whose graphical features are not available to the model) and statutes (existing nodes). Extensive experiments on the dataset show that our model comfortably outperforms several state-of-the-art baselines, by exploiting the graphical structure along with textual features.

1 Introduction

The task of Legal Statute Identification (LSI) is important in the judicial field, and involves identifying the possible set of *statutory laws* that are relevant, or might have been violated, given the *natural language description of the facts of a situation*. This task needs to be performed at different stages of litigation by different experts, including police personnel, lawyers and judges. An automated system for LSI can greatly promote access to Law by the common masses.

Existing LSI methods and their limitations: Early methods for LSI modeled the task using statistical or simple machine learning methods (Kort 1957; Lauderdale and Clark 2012). Recently, several neural architectures have modeled the problem as a text classification task, attempting to extract increasingly richer features from the text (Luo et al. 2017; Wang et al. 2018, 2019; Xu et al. 2020). Some of these methods simplify the task by trying to identify only the most relevant statute, no longer retaining its multi-label nature.

Historically maintained court case data have been popularly used to create datasets to train the LSI models (Chalkidis, Androutsopoulos, and Aletras 2019; Xiao et al. 2018). However, almost all existing methods for this task rely only on the text of the facts (and statutes) to perform the classification task. The existing methods do *not utilize the legal statute citation network* between court case documents and statutes, that has been shown to be a rich source of legal knowledge that is useful for tasks such as estimating legal document similarity (Bhattacharya et al. 2020). However, there has not been any attempt to utilize the legal citation network for the LSI task.

Graph formulation of LSI: This work is the first attempt towards LSI by utilizing both the textual content of statutes/-facts and a *heterogeneous statute citation network*. In this network, statutes and document are nodes of different types, and citation links exist between these nodes *iff* a particular section is cited by a particular document. Additionally, statute nodes may also be connected via defined hierarchies specified in the written law itself. Given such a network, the task of LSI boils down to predicting the existence of links between statute-nodes and *newly entering* document-nodes.

However, most existing supervised network-based training methods for link prediction *do not work well for out-of-sample nodes*, or nodes that the model has not seen during training. Indeed, in our case, we can only provide the document nodes and their citation links during training time. While testing, every document-node is previously unseen to the model, and the model must predict whether links exist between a new document-node and the statute nodes. Thus, approaches that seek to learn effective node representations for link prediction will fail during test time in our setting. Rather, one has to capitalize on the fact that two kinds of information are available at all times — the texts corresponding to each statute/document, and the statute nodes themselves (and hierarchical links between them).

We, therefore, formulate the LSI problem as an **inductive link prediction** task (Hao et al. 2020) on the heterogeneous citation network, between the previously seen Section nodes and a new, unknown fact node. We use a hybrid learning mechanism to learn two kinds of representations for each node – attribute (generated from text) and structural (generated from network). By forcing the model to learn attribute representations that closely match structural representations of the same node, we ensure that the model can generalize better by generating more robust, feature-rich attribute representations for unseen document nodes during test time.

A new LSI dataset: There do not exist many good quality datasets for LSI in English. The ECHR dataset (Chalkidis, Androutsopoulos, and Aletras 2019) is considerably small ($\sim 11.5K$ documents/facts) and does not provide the text of the statutes. Another popular dataset, CAIL (Xiao et al. 2018) is much larger; it is, however, in the Chinese language. Importantly, the *average no. of statutes cited per document* is quite low in both these datasets (0.71 for ECHR and 1.09 for CAIL), i.e., most documents cite at most one statute (or none in the case of ECHR). However, the LSI problem is inherently multi-label, and these datasets do not reflect the true multi-label nature of the LSI problem.

We construct a new, fairly large dataset ($\sim 66K$ docs) from case documents and statutes (in English) from the Indian judiciary. This dataset, which we call the *Indian Legal Statute Identification* (ILSI) dataset, captures the multi-label nature of the LSI problem more truly (details in Section 6).

Contributions of present work: To sum up, our contributions are as follows: (1) To our knowledge, we are the first to use the statute citation network in conjunction with textual descriptions for the task of Legal Statute Identification. We propose a novel architecture **LeSICiN (Legal Statute Identification using Citation Network)** for the task, that outperforms several state-of-the-art baselines for LSI (improvement of 19.2% over the closest competitor). (2) We construct a large-scale LSI dataset from Indian court case documents, where the task is to identify Sections of the Indian Penal Code, the major criminal law in India. The dataset and our codes are available at https://github.com/Law-AI/LeSICiN.

2 Related Work

Legal Statute Identification (LSI) has been widely studied by researchers in an attempt to automate the process.

Initial Efforts: The earliest LSI approaches used statistical algorithms along with hand-crafted rules (Kort 1957; Ulmer 1963; Segal 1984; Lauderdale and Clark 2012). Later, LSI was approached as a text classification problem based on manually engineered features (Liu and Hsieh 2006; Aletras et al. 2016). However, such hand-crafted rules/features do not allow these methods to generalize well.

Neural models for LSI: Recently developed attention-based neural models for LSI look to extract richer features from text using techniques such as dynamic context vectors (Luo et al. 2017), dynamic thresholding (Wang et al. 2018), hierarchical classification (Wang et al. 2019) or pre-trained BERT-based contextualizers (Chalkidis, Androutsopoulos, and Aletras 2019). Xu et al. (2020) split statutes into communities and use a novel graph distillation operator to identify intra-community features; however, they do not use the citation network in any way. We use all the above-mentioned methods as baselines in this paper.

LJP and Multi-task approaches: Legal Judgment Prediction (LJP) is an umbrella problem that encompasses several related sub-tasks such as identifying statutes (LSI), charges, the term of penalty, etc. A different class of algorithms attempt to exploit the relationship between these multiple related tasks (of LJP) by modeling them together (Zhong et al. 2018; Yang et al. 2019). These models work effectively only when training data is available for more than one tasks. Since our main focus is LSI, and we do not have training data for other tasks such as predicting the term of penalty, we do not consider these as baselines in this work.

Chapter	Topic	Section
Offences affecting Human Body	Offences affecting Life	**299:** Culpable homicide
		307: Attempt to murder
	Hurt	**321:** Voluntarily causing hurt
		334: Voluntarily causing hurt on provocation
Offences against Property	Robbery and Dacoity	**390:** Robbery
		396: Dacoity with murder
	Criminal Trespass	**441:** Criminal trespass
		446: House breaking by night

Table 1: An illustrative part of the hierarchy of the Indian Penal Code (IPC) Act divided into Chapters, Topics and Sections (which are mostly cited by documents).

3 Data Preparation

We develop a novel dataset for the Legal Statute Identification (LSI) task using criminal case documents and statutes from the Indian judiciary. This section describes the dataset.

Statutes: In Indian Law, most criminal offences are described in the Indian Penal Code (IPC), which is an *Act*. The IPC *Act* has a hierarchical structure – the *Act* is divided into coarse-grained categories called *Chapters*, which are further subdivided into fine-grained categories called *Topics*. Each Topic groups together a set of *Sections* that are based on the same crime. Sections are statutory legal articles that are usually cited from case documents. The text of a Section describes the nature and circumstances of the crime, and litigation procedures involved. In this paper, we use the terms 'Statute' and 'Section' interchangeably. Table 1 shows a part of the Indian Penal Code, depicting examples of Chapters, Topics and Sections.

Facts: We collected $\sim 100K$ court case documents from the Supreme Court and six major High Courts in India, from the website https://indiankanoon.org. We collected only documents that cite at least one Section from the IPC. Case documents are comprised of many semantic parts, such as the *facts*, *arguments*, *ruling*, etc. (Bhattacharya et al. 2019). Since the input to the LSI task consists of only the *facts* (that led to filing of the case), we used the Hier-BiLSTM-CRF classifier developed by Bhattacharya et al. (2019) to extract the facts from the collected case documents (F1 of 0.839 over facts).

The ILSI dataset: The IPC contains more than 500 Sections, but a large majority of them are seldom cited. Hence, we chose to focus on the 100 most frequently cited Sections of IPC as the set of labels in our dataset. We consider the facts from only those case documents that cite at least one of these top 100 Sections, and we end up with 66,090 such facts. Table 2 (last column) shows the basic statistics of the dataset developed in this work, which we call *Indian Legal Statute Identification* (ILSI) dataset. Note that we mask

Dataset	**ECHR**	**CAIL**	**ILSI**
Language	English	Chinese	English
No. of documents/facts	11,478	2,676,075	66,090
No. of labels/statutes	66	183	100
Avg. no. of words per doc	2406	1444	1232
Avg. no. of labels per doc in test set	0.71	1.09	**3.78**
Statute/Label Text	No	Yes	Yes

Table 2: Statistics of the new ILSI dataset and comparison with two other LSI datasets (ECHR and CAIL). ILSI reflects the multi-label nature of the LSI task more closely.

named entities in the text; this is a common preprocessing step in Legal NLP, meant to minimize demographic bias in models (Chalkidis, Androutsopoulos, and Aletras 2019).

We split the dataset into *train*, *validation* and *test* parts in the ratio of $64 : 16 : 20$. Thus, we have $42,884$ training documents, $10,203$ validation documents, and $13,043$ test documents. Since this is a multi-label classification dataset, we ensure that the distribution of labels is balanced across all three sets, using *iterative stratification* (Sechidis, Tsoumakas, and Vlahavas 2011).

Each input to the LSI task is the textual description of a fact (obtained from a particular case document). The set of Sections cited in the said case document is considered the gold-standard set of labels for the given fact. The ILSI dataset is available at https://github.com/Law-AI/LeSICiN.

Comparison of ILSI with existing LSI datasets: Table 2 compares ILSI with two popular datasets for the task of LSI, namely, ECHR (Chalkidis, Androutsopoulos, and Aletras 2019) and CAIL (Xiao et al. 2018). The ECHR dataset is relatively small, and contains many documents which do not cite any statute/label (since the dataset was designed for both binary and multi-label classification). Hence, the average number of labels per document is less than 1. Also, notably, the ECHR dataset does *not contain any text for the statutes/labels*. The CAIL dataset (in the Chinese language) is quite large and includes the text of the statutes. But the average number of labels per document is very close to 1 (1.09), indicating that a large fraction of documents do not cite more than one label. Whereas, for ILSI, the average number of labels per document is relatively high (3.78), since a significant fraction of the documents cite more than one labels. Hence, the ILSI dataset reflects more truly the multi-label nature of the LSI problem.

4 Formalization of the Problem

This section discusses the standard formulation of the LSI task as a multi-label classification problem, and how we formulate LSI as a link prediction task over a graph.

Multi-label Classification Formulation: Let $F = \{f_1, f_2, \ldots, f_{|F|}\}$ denote the entire set of fact descriptions (documents), and $S = \{s_1, s_2, \ldots, s_{|S|}\}$ denote the set of IPC Sections (labels). Since more than one sections may be relevant to a fact f, each instance is denoted as a tuple $\langle f, \mathbf{y}_f \rangle$. Here, $\mathbf{y}_f \in \{0,1\}^{|S|}$, where $\mathbf{y}_f[s] \in \{0,1\}$ indicates whether Section s is relevant to fact f.

The LSI task requires us to develop a function $\mathcal{F}(\cdot)$ such that $\mathcal{F}(f, S) = \hat{\mathbf{y}_f}$; where $\hat{\mathbf{y}_f} \in \{0,1\}^{|S|}$, with $\hat{\mathbf{y}_f}[s] \in \{0,1\}$ denoting the function's prediction of whether Section s is relevant to fact f.

Link Prediction Formulation: We model the LSI task using a graph formalism, over a *legal citation network*. Each Section and each Fact (from training instances) is treated as a node in a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with a node mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and an edge mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{A}$ and $\mathcal{R}$ represent the types of nodes and types of relations between nodes, respectively. In our case, we have fact nodes (F) and Section nodes (S), which are accompanied with their attributes. Additionally, we have placeholder node types to denote the hierarchical levels of IPC – Topics (T), Chapters (C) and IPC Act (A) (see Section 3). There exist several types of relations between the nodes – 'cites' (*ct*) and 'cited by' (*ctb*) relationships between nodes of types F and S, 'includes' (inc) and 'part of' (po) relationships between the successive node hierarchy types A, C, T and S. Figure 1 shows a pictorial representation of this network.

We construct the network using the following steps. (i) Assign the set of nodes in each node type, e.g., A, C, T, S, F. Set $\mathcal{V} = \mathcal{V}_F \cup \mathcal{V}_S \cup \mathcal{V}_T \cup \mathcal{V}_C \cup \mathcal{V}_A$. (ii) For given nodes u and v belonging to *different, successive levels of hierarchy* in the Act (IPC) s.t. $\phi(u) \in \{A, C, T\}$ and $\phi(v) \in \{C, T, S\}$, include links $(u, v) \in \mathcal{E}_{\text{inc}}$ and $(v, u) \in \mathcal{E}_{\text{po}}$ iff v is categorized under the broader hierarchy level of u. (iii) For given nodes u and v s.t. $u \in \mathcal{V}_F$ and $v \in \mathcal{V}_S$, include links $(u, v) \in \mathcal{E}_{\text{ct}}$ ('cites' link) and $(v, u) \in \mathcal{E}_{\text{ctb}}$ ('cited by' link) iff $\mathbf{y}_u[v] = 1$.

Given a new fact f *at test time*, the task is to identify the relevant sections. This problem can be seen as **inductive link prediction** over the network described above, i.e., predicting the possibility of the existence of a link between the new fact f and the section nodes in the network. Note that inductive link prediction is the task of predicting a link between a pair of nodes, where *one or more nodes might not be visible to the model during training.*

Formally, modeling LSI as Link Prediction requires us to develop a function $\mathcal{Q}(.)$ such that $\mathcal{Q}(u, v, \mathcal{G}) = \hat{y_{uv}}$, where $u \in \mathcal{V}_F$, $v \in \mathcal{V}_S$ and $\hat{y_{uv}} \in \{0,1\}$ denotes the function's prediction of whether a link should exist between u and v.

5 Proposed LSI Model: LeSICiN

Figure 1 gives an overview of our proposed model **LeSICiN**. Each fact $f \in F$ and section $s \in S$ is accompanied by a text x_f and x_s respectively, which can be considered as attributes of the node. The citation network provides the structural information. We have two separate encoders for encoding attributes and structural information, which are shared across both facts and Sections.

We adopt the technique used by DEAL (Hao et al. 2020) to generate good quality embeddings for unseen nodes (new fact descriptions) using only their attributes at test time. During training, we obtain both attribute and structural embeddings of both node types, facts and Sections. These are combined in different ways to generate three types of scores – attribute, structural and alignment. During testing, since structural embeddings of new facts are not available, we can only generate attribute and alignment scores. The alignment score enables the model to correlate the two kinds of embeddings, for generalizing well to unseen nodes during testing.

Next, we describe the attribute and structural encoders, and the scoring function used by the architecture.

5.1 Attribute Encoder

We use a shared attribute encoder for both facts and Sections. A text portion in our formulation is a nested sequence of sentences and words. Hence we encode the text using the Hierarchical Attention Network (HAN) (Yang et al. 2016). Specifically, by feeding the fact text x_f and each Section text x_s to HAN, we obtain a single, attention-weighted representation for the entire texts namely, $\mathbf{h}_f^{(a)}$ and a set of Section representations $\{\mathbf{h}_s^{(a)} \mid s \in S\}$.

5.2 Structural Encoder

To exploit the rich, semantic information of the Citation Network, we carefully design various metapath schemas, following the process adopted by Fu et al. (2020). A *metapath* is a sequence $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_1} \ldots \xrightarrow{R_l} A_{l+1}$, which denotes the composite relation $R_1 \circ R_2 \circ \ldots \circ R_l$ between node types $A_1, A_2, \ldots, A_{l+1}$. A metapath only defines a schema; there may be multiple sequence of nodes originating from the same starting node, following the same metapath schema P. Each such sequence is called a metapath instance of P. The P metapath-based neighbourhood of a node v, denoted as $\mathcal{N}_v^P$ is defined as all the nodes that can be reached from v using the schema P. Any neighbour connected by two or more metapath instances is represented as two or more different nodes in $\mathcal{N}_v^P$.

To extract the structural information from the citation network, we make use of different metapath schemas with different node types as the start nodes. For example, for nodes of type F (facts) we have a metapath schema $F \xrightarrow{ct} S \xrightarrow{ctb} F$, capturing the type of relationship that exists between *facts that cite the same Section*. From Figure 1, we can observe multiple instances of this metapath schema, such as $F1 - S1 - F3$ and $F2 - S3 - F3$. As another example, for nodes of type S (sections), we have a metapath schema $S \xrightarrow{po} T \xrightarrow{po} C \xrightarrow{inc} T \xrightarrow{inc} S$, capturing the relationship between *Sections defined under the same Chapter*. From Figure 1, we can see that $S1 - T1 - C2 - T2 - S3$ and $S1 - T1 - C2 - T1 - S2$ are two instances of this schema. We use a total of 4 fact-side and 4 Section-side metapath schemes, which are listed in the **supplementary material**.

Next, we describe the individual node representation, followed by intra- and inter-metapath aggregation to obtain the structural embedding.

Node Embedding: We have a set of parametric node embedding matrices $\{\mathbf{X}_A \mid A \in \mathcal{A}\}$ that are used to initialize each node v with its feature vector $\mathbf{x}_v$. Since the initial feature vectors might be of different dimensions and may map to different latent spaces, we need to transform them to the same dimensionality and space. For a node $v \in \mathcal{V}_A$ for $A \in \mathcal{A}$, we have $\mathbf{x}_v = \mathbf{X}_A \cdot \mathbf{I}_v$ and $\mathbf{h'}_v = \mathbf{W}_A \cdot \mathbf{x}_v$. Here $\mathbf{X}_A \in \mathbb{R}^{d_A \times |\mathcal{V}_A|}$ is the node embedding matrix for nodes of type $A \in \mathcal{A}$; $\mathbf{I}_v \in \mathbb{R}^{|\mathcal{V}_A|}$ is the one-hot identity vector for node v; and $\mathbf{x}_v \in \mathbb{R}^{d_A}$ is the initial feature vector for v. Further, $\mathbf{W}_A \in \mathbb{R}^{d' \times d_A}$ is the parametric transformation matrix for node type $A \in \mathcal{A}$; and $\mathbf{h'}_v \in \mathbb{R}^{d'}$ is the transformed latent vector of uniform dimensionality (across node types). After transformation, all node embeddings are ready to be fed to the aggregation architecture.

Intra-Metapath Aggregation: First, we need to aggregate all the node embeddings for a target node v under a particular metapath schema P. For a metapath instance $P(v, u)$ connecting v with its metapath-based neighbour $u \in \mathcal{N}_v^P$, we use a metapath instance encoder $g_\theta(.)$ to generate a representation for the instance $P(v, u)$. We adopt the relational rotation encoder used by Fu et al. (2020).

Consider that $P(v, u) = \{n_0, n_1, \ldots, n_M\}$, with $n_0 = u$ and $n_M = v$. We thus have

$$\mathbf{q}_i = \mathbf{h'}_{n_i} + \mathbf{q}_{i-1} \odot \mathbf{r}_i; \quad \mathbf{h}_{P(v,u)} = \frac{\mathbf{q}_M}{M+1}$$

where $\mathbf{q}_0 = \mathbf{h'}_u$, $\mathbf{r}_i \in \mathbb{R}^{d'}$ is the learned relation vector for $R_i \in \mathcal{R}$ and $\mathbf{h}_{P(u,v)}$ is the vector representation of the metapath instance $P(u, v)$. Now, we have obtained representations $\mathbf{h}_{P(u,v)}$ for each $u \in \mathcal{N}_v^P$. We attentively combine these P-based representations for the target node v as

$$e_{vu}^P = \text{LeakyReLU}\left(\mathbf{a}_P^\intercal \cdot [\mathbf{h'}_v || \mathbf{h}_{P(v,u)}]\right)$$

$$\alpha_{vu}^P = \text{Softmax}_{u \in \mathcal{N}_v^P}\left(e_{vu}^P\right)$$

$$\mathbf{h}_v^P = \text{ReLU}\left(\sum_{u \in \mathcal{N}_v^P} \alpha_{vu}^P \cdot \mathbf{h}_{P(v,u)}\right)$$

where $\mathbf{a}_P \in \mathbb{R}^{2d'}$ is the parameterized context vector and $\mathbf{h}_v^P \in \mathbb{R}^{d'}$ is the aggregated representation from all metapath neighbours of v in P.

Inter-Metapath Aggregation: Now, we need to aggregate the metapaths across different schemas. Consider $\mathcal{P}_A = \{P_1, P_2, \ldots, P_N\}$ as the set of metapath schemas which start with node type $A \in \mathcal{A}$. For a node $v \in \mathcal{V}_A$ of type A, we have a set of N representations $\{\mathbf{h}_v^{P_i}, \forall i \in [1, N]\}$.

First, information for each metapath schema is summarized across all nodes $v \in \mathcal{V}_A$ as

$$\mathbf{s}_{P_i} = \frac{1}{|\mathcal{V}_A|} \sum_{v \in \mathcal{V}_A} \tanh\left(\mathbf{M}_A \cdot \mathbf{h}_v^{P_i} + \mathbf{b}_A\right)$$

where $\mathbf{M}_A \in \mathbb{R}^{d_m \times d'}$ and $\mathbf{b}_A \in \mathbb{R}^{d_m}$ are learnable parameters. Then, attention mechanism is employed to aggregate all the M representations as $e_{P_i} = \mathbf{q}_A^\top \cdot \mathbf{s}_{P_i}$,

$$\beta_{P_i} = \text{Softmax}_{P_i \in \mathcal{P}_A}\left(e_{P_i}\right) \quad \mathbf{h}_v = \sum_{P_i \in \mathcal{P}_A} \beta_{P_i} \mathbf{h}_v^{P_i}$$

where $\mathbf{q}_A \in \mathbb{R}^{d_m}$ is the parameterized context vector.

Thus, at the end of the structural encoding phase, we get a single representation of the fact $\mathbf{h}_f^{(s)}$ and a set of representations for each section $\{\mathbf{h}_s^{(s)} \mid s \in S\}$.

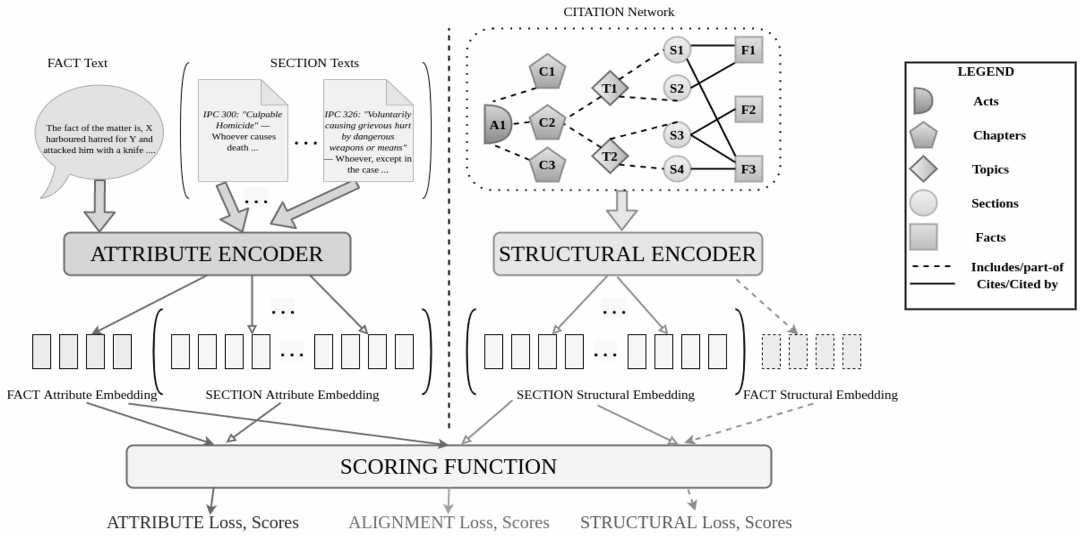

CITATION Network
FACT Text
SECTION Texts
The fact of the matter is, X harboured hatred for Y and attacked him with a knife ....
IPC 300: "Culpable Homicide" — Whoever causes death ...
IPC 326: "Voluntarily causing grievous hurt by dangerous weapons or means" — Whoever, except in the case ...
A1
C1
C2
C3
T1
T2
S1
S2
S3
S4
F1
F2
F3
LEGEND
Acts
Chapters
Topics
Sections
Facts
Includes/part-of
Cites/Cited by
ATTRIBUTE ENCODER
STRUCTURAL ENCODER
FACT Attribute Embedding
SECTION Attribute Embedding
SECTION Structural Embedding
FACT Structural Embedding
SCORING FUNCTION
ATTRIBUTE Loss, Scores
ALIGNMENT Loss, Scores
STRUCTURAL Loss, Scores

Figure 1: Architecture of our proposed model **LeSICiN**. The **attribute encoder** is used for converting text to vector representations, and the **structural encoder** is used to generate representations for Sections and training documents. These representations are combined and fed to the **scoring mechanism** to generate losses and scores.

5.3 Scoring Mechanism

We design a scoring function $m_\theta(f; \{s \mid s \in S\})$ to assign a score to each Section $s \in S$ for fact f. Leveraging the fact that Sections in IPC do have a defined sequential order and related semantics, we first use an LSTM to contextualize these embeddings $\mathbf{h}_s$ for $s \in S$ as $\tilde{\mathbf{h}}_s = \text{Bi-LSTM}\left(\mathbf{h}_s; [\mathbf{h}_t \mid t \in S]\right)$. Then, we use the standard attention mechanism to generate a single aggregated embedding $\mathbf{h}_S$ representing the set S:

$$e_s = \mathbf{w}_S^\top \cdot \tanh\left(\mathbf{M}_S \cdot \tilde{\mathbf{h}}_\mathbf{s} + \mathbf{b}_S\right)$$

$$\gamma_s = \text{Softmax}_{s\in S}\left(e_s\right) \quad \mathbf{h}_S = \sum_{s\in S} \gamma_s \tilde{\mathbf{h}}_s$$

where $\mathbf{w}_S \in \mathbb{R}^{d_s}$ is the parameterized context vector and $\mathbf{M}_S \in \mathbb{R}^{d_s \times d'}$ and $\mathbf{b}_S \in \mathbb{R}^{d_s}$ are learnable parameters.

Finally, we generate the score for each class as:

$$\mathbf{o}_f = m_\theta(f; \{s \mid s \in S\}) = \sigma\left(\mathbf{W}_C \cdot [\mathbf{h}_f || \mathbf{h}_S] + \mathbf{b}_C\right)$$

where $\mathbf{W}_C \in \mathbb{R}^{|S| \times 2d'}$ and $\mathbf{b}_C \in \mathbb{R}^{|S|}$ are the learnable parameters for the final classification layer.

Since we have two sets of embeddings for each fact f and section c_i, we can generate three sets of scores, namely –

(i) **Attribute Score:** $\mathbf{o}_f^{(a)} = m_\theta(\mathbf{h}_f^{(a)}; \{\mathbf{h}_s^{(a)} \mid s \in S\})$ matches the attribute embedding of facts with Sections;

(i) **Structural Score:** $\mathbf{o}_f^{(s)} = m_\theta(\mathbf{h}_f^{(s)}; \{\mathbf{h}_s^{(s)} \mid s \in S\})$ matches the structural embedding of facts with Sections;

(iii) **Alignment Score:** $\mathbf{o}_f^{(l)} = m_\theta(\mathbf{h}_f^{(a)}; \{\mathbf{h}_s^{(s)} \mid s \in S\})$ matches the attribute embedding of facts with structural embedding of Sections.

The structural score can only be calculated during training time, since the graphical structure of the facts at test/inference time is not available. During training, the structural score helps the model to understand the graphical structure of sections and training documents. However, the graphical structure of the sections is available at all times, and thus the alignment score actually tunes the model to generate similar attribute and structural representations.

Using Dynamic Context: To provide better guidance to the attention mechanism for the structural encoder through the attribute embeddings, we replace the static context vectors in the structural encoder with dynamically generated vectors from the attribute embeddings of the same node (Luo et al. 2017). In the scoring function, we use the fact embeddings to generate the dynamic context.

$$\mathbf{a}_P = \mathbf{T}_P \cdot \mathbf{h}_v^{(a)}; \quad \mathbf{q}_A = \mathbf{T}_A \cdot \mathbf{h}_v^{(a)}; \quad \mathbf{w}_S = \mathbf{T}_S \cdot \mathbf{h}_f$$

where $\mathbf{T}_P \in \mathbb{R}^{2d' \times d'}$, $\mathbf{T}_A \in \mathbb{R}^{d' \times d'}$ and $\mathbf{T}_S \in \mathbb{R}^{d' \times d'}$ are the learnable transformation matrices.

5.4 Training and Prediction

The loss function has three parts, analogous to the three scores, namely, $\mathcal{L}^{(a)}$, $\mathcal{L}^{(s)}$ and $\mathcal{L}^{(l)}$. We use weighted Binary Cross Entropy Loss to calculate each component as:

$$l_s^{(t)} = w_s \mathbf{y}_f[s] \log\left(\mathbf{o}_f^{(t)}[s]\right) + (1 - \mathbf{y}_f[s]) \log\left(1 - \mathbf{o}_f^{(t)}[s]\right)$$

$$\mathcal{L}^{(t)} = -\frac{1}{|B|} \sum_{f\in B} \sum_{s\in S} l_s^{(t)}; \quad t \in \{a, s, l\}, s \in S$$

where w_s denotes the weight of each positive sample of class $s \in S$ and B denotes a mini-batch of facts.

The *vanilla weighting scheme* (VWS) assigns $w_s = N/f_s, \forall s \in S$, where N is the total no. of training documents and f_s is the no. of training documents that cite s. However, this scheme can sometimes lead to very large weights for the rare labels $f_s << N$. To address this issue, we propose a *threshold-based weighting scheme* (TWS) defined as $w_s = \min\left(f_{max}/f_s, \eta\right)$ where f_{max} is the frequency of the most cited label, and η is a threshold value determined on the validation set. This scheme caps the class weights at a reasonable value, so that the model does not compromise performance on the frequent classes for minor improvements over the rare ones.

We have the final loss as: $\mathcal{L} = \theta_a \mathcal{L}^{(a)} + \theta_s \mathcal{L}^{(s)} + \theta_l \mathcal{L}^{(l)}$ During test time, the structural score is not available. Thus, $\hat{\mathbf{y}_f} = I\left(\lambda_a \mathbf{o}_f^{(a)} + \lambda_l \mathbf{o}_f^{(l)} \geq \tau\right)$ where $\hat{\mathbf{y}_f} \in \{0,1\}^{|S|}$ and $\hat{\mathbf{y}_f}[s] \in \{0,1\}$ indicates whether the model predicts Section s to be relevant to fact f or not, and τ is the threshold value set using the validation set.

6 Experiments & Results

In this section, we compare the performance of our model with that of several baselines. We also analyze the effectiveness of our model and its various components.

Baselines: We consider the following state-of-the-art baselines for ILSI: **(1) FLA** (Luo et al. 2017) uses two HANs to generate fact embeddings and Section embeddings **(2) DPAM** (Wang et al. 2018) learns the distribution of *pairs of Sections* instead of individual Sections, for better learning the rare Sections; **(3) HMN** (Wang et al. 2019) models the task as a hierarchical classification task with two levels (analogous to *Acts* and *Sections* in our work); **(4) LADAN** (Xu et al. 2020) groups the Sections into non-overlapping communities based on TF-IDF similarity measures before performing Graph distillation on each community **(5) HBERT** (Chalkidis, Androutsopoulos, and Aletras 2019) uses BERT (Devlin et al. 2018) to generate embeddings for each sentence of the Fact, and an LSTM-Attn layer on top to generate the final embedding for each document before classifying. Note that this model *does not utilize the text of the statutes*; **(6) HLegalBERT:** A variation of HBERT with the same architecture, but using LegalBERT (Chalkidis et al. 2020) to generate domain-aware embeddings for each sentence; **(7) DEAL:** The base DEAL (Hao et al. 2020) using a node embedding matrix as the structural encoder, cosine similarity for scoring links, and a network distance-based weighting of the negative samples for the loss function.

Hyperparameters: We apply the same settings to all competing methods to ensure fair competition. Every model is trained and validated on the train and validation sets described in Section 3, for 100 epochs. At the end of training, the model state yielding the best validation result is used over the test set. Some hyper-parameters are also fixed across all models – (i) embedding dimension of 200 in all cases except 768 for HBERT, (ii) Adam Optimizer with learning rate in the range $[0.01, 0.000001]$, (iii) dropout probability of 0.5, and (iv) batch size of 32.

For model-specific configurations required in LeSICiN, we sample 8 metapath instances per schema, per node. To place extra emphasis on network learning, we set $\theta_a = 1$, $\theta_s = 2$, $\theta_l = 3$ and $\lambda_a = 0.25$, $\lambda_l = 0.75$. Based on validation set performance, we set $\tau = 0.65$ and $\eta = 10.0$.

To improve the models' understanding of legal text, we use sent2vec (Pagliardini, Gupta, and Jaggi 2018) embeddings of dimension 200 trained on a large legal corpus of Indian case documents, to to initialize the sentence embeddings of all models (except HBERT and HLegalBERT which have their own encoding methods).

We have used the source codes for the baselines where available. We have coded our model and other models (if code unavailable) in PyTorch (Paszke et al. 2019). For the network part, we use PyTorch Geometric (Fey and Lenssen 2019). We trained all models on an NVIDIA Tesla T4 with 16 GB GPU memory.

Evaluation Metrics: Since the label distribution (frequency of statutes being cited) is very skewed, we use macro-averaged versions (i) *macro-Precision*, (ii) *macro-Recall* and (iii) *macro-F1*. Also, we use (iv) the Jaccard overlap between the gold standard set of labels and the predicted set of labels (of a test Fact). All metrics are averaged over the entire test dataset.

6.1 Comparative Results

Table 3 shows the performance of all competing models on the test set. In terms of macro-F1, our proposed model LeSICiN performs the best across all methods, followed by DPAM and FLA. The last column of Table 3 shows a configuration of LeSICiN tuned for higher recall, by reducing the classification threshold to $\tau = 0.3$. This is for the sake of comparison with the best configuration of LeSICiN (second-last column), which has been tuned for highest F1 with $\tau = 0.65$.

In the case of FLA, the fact-aware Section representations seem to give good performance, especially in terms of Recall (highest macro-Recall across all methods). Methods like HMN and LADAN that perform two-step classification do not perform well, possibly since performance of the second step depends crucially on that of the first step; the distribution of Sections over Topics (in case of HMN) and Sections over communities (for LADAN) might be too skewed, leading to low Precision values (and hence low F-score) for both models. The BERT-based HBERT and HLegalBERT do not perform well at all, most possibly because these models do not make use of the Section texts (since we follow the HBERT implementation in (Chalkidis, Androutsopoulos, and Aletras 2019)). DEAL uses the graph structure to extract valuable distinguishing features, achieving the second highest macro-Recall after FLA. However, it does not make use of the graph heterogeneity nor complex GNNs, just a simple node embedding lookup. Thus its overall performance (macro-F1) is relatively low.

Interestingly, several models suffer from the problem of low Precision and high Recall, demonstrating a significant difference between macro-P and macro-R, except for DPAM and LeSICiN, which are actually the top performing methods. However, as the last column in Table 3 shows, LeSICiN can be tuned to yield higher recall than all baselines with slight sacrifice in F1. LeSICiN with $\tau = 0.3$ performs similar to FLA in terms of all the metrics.

6.2 Analysis of Our Model

Ablation experiments: We now try the following various variations of LeSICiN, to understand the effectiveness of different components of LeSICiN:

Metric	FLA	DPAM	HMN	LADAN	HBERT	HLegalBERT	DEAL	LeSICiN	LeSICiN ($\tau = 0.3$)
Macro-P	12.19	27.11	10.27	12.54	5.79	4.36	12.66	**27.90**	11.99
Macro-R	69.25	27.43	57.30	46.17	51.43	52.55	64.83	31.32	**69.42**
Macro-F1	19.60	23.86	16.12	17.93	7.91	7.58	19.43	**28.45**	19.77
Jaccard	11.64	14.44	9.56	10.28	4.47	4.01	11.44	**17.57**	11.49

Table 3: Comparative results of the baselines and the proposed LeSICiN on the test set. Last column shows a variations of LeSICiN, that optimizes for Recall. All values are in percentages. Best value for each metric is in boldface. Differences in Macro-F1 between LeSICiN and all baselines are all statistically significant (paired t-Test with 95% confidence).

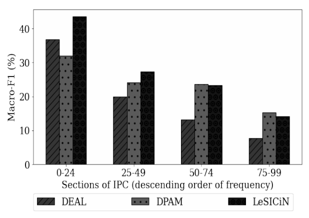
Model	Macro-P	Macro-R	Macro-F1	Jacc
LeSICiN	27.90	31.32	28.45	17.57
LeSICiN-E	27.87	26.99	25.18	15.38
LeSICiN-B	21.53	36.04	24.66	14.89
LeSICiN-S	22.00	43.19	25.40	15.32
LeSICiN-V	19.50	47.60	25.52	15.48
LeSICiN-SG	26.50	29.80	26.90	16.17

Table 4: Ablation test results on our proposed model LeSICiN on the test set. All values are in percentages.

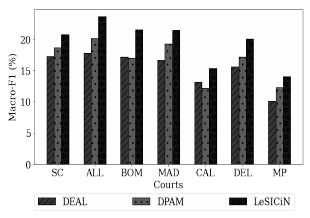
(1) *LeSICiN-E:* To inspect the graph encoding capabilities of LeSICiN, we replace the metapath-based aggregation network with a simple node embedding lookup table, similar to what is used in DEAL.
(2) *LeSICiN-B:* We replace the Scoring Function described in Section 5.3 with a simple Bilinear layer, i.e., for each $\langle Fact, Sec \rangle$ pair, this variation generates a single score that denotes how relevant the members of the pair are.
(3) *LeSICiN-S:* We remove the structural loss $\mathcal{L}^{(s)}$ calculated based on Section structural embeddings and Fact attribute embeddings during training.
(4) *LeSICiN-V:* We use VWS for weighting loss instead of our proposed TWS (described in Section 5.4).
(5) *LeSICiN-SG:* To investigate the gains of modeling heterogeneity in the citation network, we replace the metapath-based aggregation network with a homogeneous graph encoder, GraphSAGE (Hamilton et al. 2017).

Table 4 shows that removal of any component from LeSICiN degrades its performance. The node embedding matrix of LeSICiN-E is not capable of capturing the complex semantic relationships in the heterogeneous citation graph and hence fails to generalize well to the test docs. The scoring function used in LeSICiN seems to have a higher impact on performance, since replacing it with a bilinear matching function (in LeSICiN-B) deteriorates macro-F1 even further. The bilinear function cannot model relationships between labels efficiently since it scores each label independently. LeSICiN-S, though inferior to LeSICiN, outperforms several baselines, probably due to the fact that the alignment loss captures the section-side aspect of the citation network. Using threshold-based weights also seem to improve performance, since LeSICiN outperforms LeSICiN-V with the vanilla weighting scheme. Finally, LeSICiN-SG does not perform as good as LeSICiN since it neglects the heterogeneity of the citation network. Importantly, despite significant ablations, the variations of LeSICiN achieve higher macro-F1 than the baselines in Table 3.

Performance on labels of varying frequencies: We check how well LeSICiN and the two closest baselines (DPAM and DEAL) perform across labels/Sections having different frequencies of citation. Figure 2a shows the performances



(a) Label frequency



(b) Across Law courts

Figure 2: Performance (macro-F1) of LeSICiN and two closest competitors DPAM and DEAL across (a) labels (statutes) of varying frequencies, (b) facts from different Law courts.

(Macro-F1 scores), where the X-axis shows the Sections sorted in decreasing order of their frequency of being cited (by the documents in the test set), and grouped into four groups. The performances of all models degrade sharply over the less frequently cited labels/Sections. LeSICiN performs much better than the baselines for the first two groups of Sections, i.e., for the 50 most-frequently cited Sections. For the less frequently cited Sections, LeSICiN performs much better than DEAL and very close to DPAM (difference between LeSICiN and DPAM is *not* statistically significant by paired t-Test with 95% confidence).

Performance on facts from different courts of Law: As discussed in Section 3, the ILSI dataset has facts extracted from case documents of *seven* Indian courts – the Supreme Court and six major High Courts (Allahabad, Bombay, Madras, Calcutta, Delhi and Madhya Pradesh). It is important for models to be generalizable across different courts, since writing styles may vary significantly across courts.

Figure 2b shows the performance of LeSICiN and the two closest competitors (DPAM and DEAL) over the facts of each Court. LeSICiN consistently performs much better than both DPAM and DEAL, across documents from all courts, thus showing its generalizability over writing styles.

7 Conclusion

We propose a novel model LeSICiN for the Legal Statute Identification (LSI) task, that utilizes the citation network of the training data to produce robust, feature-rich representations for the text, which generalize well to unseen test documents, across a range of different parameters. We also introduce a new, large-scale English dataset for LSI constructed from Indian judiciary documents.

Acknowledgements: The research is partially supported by TCG CREST through the project "Smart Legal Consultant: AI-based Legal Analytics". S. Paul is supported by the Prime Minister's Research Fellowship (PMRF) from the Government of India.

References

Aletras, N.; Tsarapatsanis, D.; Preoţiuc-Pietro, D.; and Lampos, V. 2016. Predicting judicial decisions of the European Court of Human Rights: A natural language processing perspective. *PeerJ Computer Science*, 2: e93.

Bhattacharya, P.; Ghosh, K.; Pal, A.; and Ghosh, S. 2020. Hier-SPCNet: A Legal Statute Hierarchy-Based Heterogeneous Network for Computing Legal Case Document Similarity. In *Proc. ACM SIGIR Conference on Research and Development in Information Retrieval*, 1657–1660.

Bhattacharya, P.; Paul, S.; Ghosh, K.; Ghosh, S.; and Wyner, A. 2019. Identification of Rhetorical Roles of Sentences in Indian Legal Judgments. In *Proceedings of JURIX*.

Chalkidis, I.; Androutsopoulos, I.; and Aletras, N. 2019. Neural Legal Judgment Prediction in English. In *Proceedings of ACL*.

Chalkidis, I.; Fergadiotis, M.; Malakasiotis, P.; Aletras, N.; and Androutsopoulos, I. 2020. LEGAL-BERT: The Muppets straight out of Law School. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2898–2904.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, 2331–2341.

Hamilton, W. L.; et al. 2017. Inductive Representation Learning on Large Graphs. NIPS'17.

Hao, Y.; Cao, X.; Fang, Y.; Xie, X.; and Wang, S. 2020. Inductive Link Prediction for Nodes Having Only Attribute Information. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 1209–1215. International Joint Conferences on Artificial Intelligence Organization.

Kort, F. 1957. Predicting Supreme Court decisions mathematically: A quantitative analysis of the "right to counsel" cases. *American Political Science Review*, 51(1): 1–12.

Lauderdale, B. E.; and Clark, T. S. 2012. The Supreme Court's many median justices. *American Political Science Review*, 106(4): 847–866.

Liu, C.-L.; and Hsieh, C.-D. 2006. Exploring phrase-based classification of judicial documents for criminal charges in chinese. In *International Symposium on Methodologies for Intelligent Systems*, 681–690. Springer.

Luo, B.; Feng, Y.; Xu, J.; Zhang, X.; and Zhao, D. 2017. Learning to predict charges for criminal cases with legal basis. In *Proceedings of EMNLP*, 2727–2736.

Pagliardini, M.; Gupta, P.; and Jaggi, M. 2018. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 528–540.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Sechidis, K.; Tsoumakas, G.; and Vlahavas, I. 2011. On the stratification of multi-label data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 145–158. Springer.

Segal, J. A. 1984. Predicting Supreme Court cases probabilistically: The search and seizure cases, 1962-1981. *American Political Science Review*, 78(4): 891–900.

Ulmer, S. S. 1963. Quantitative analysis of judicial processes: Some practical and theoretical applications. *Law & Contemp. Probs.*, 28: 164.

Wang, P.; Fan, Y.; Niu, S.; Yang, Z.; Zhang, Y.; and Guo, J. 2019. Hierarchical Matching Network for Crime Classification. In *Proceedings of SIGIR*, 325–334. ACM.

Wang, P.; Yang, Z.; Niu, S.; Zhang, Y.; Zhang, L.; and Niu, S. 2018. Modeling dynamic pairwise attention for crime classification over legal articles. In *Proceedings of SIGIR*, 485–494. ACM.

Xiao, C.; Zhong, H.; Guo, Z.; Tu, C.; Liu, Z.; Sun, M.; Feng, Y.; Han, X.; Hu, Z.; Wang, H.; et al. 2018. Cail2018: A large-scale legal dataset for judgment prediction. *arXiv preprint arXiv:1807.02478*.

Xu, N.; Wang, P.; Chen, L.; Pan, L.; Wang, X.; and Zhao, J. 2020. Distinguish Confusing Law Articles for Legal Judgment Prediction. In *Proceedings of ACL*.

Yang, W.; Jia, W.; Zhou, X.; and Luo, Y. 2019. Legal judgment prediction via multi-perspective bi-feedback network. In *Proceedings of IJCAI*.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL*, 1480–1489.

Zhong, H.; Guo, Z.; Tu, C.; Xiao, C.; Liu, Z.; and Sun, M. 2018. Legal judgment prediction via topological learning. In *Proceedings of EMNLP*, 3540–3549.