Journal of Data Intelligence, Vol. 3, No. 3 (2022) 350–365

LEGAL PARTY EXTRACTION FROM LEGAL OPINION TEXTS USING RECURRENT DEEP NEURAL NETWORKS

CHAMODI SAMARAWICKRAMA
Department of Computer Science & Engineering, University of Moratuwa, Sri Lanka
chamodii.16@cse.mrt.ac.lk

MELONIE DE ALMEIDA
Department of Computer Science & Engineering, University of Moratuwa, Sri Lanka
melonie.16@cse.mrt.ac.lk

NISANSA DE SILVA
Department of Computer Science & Engineering, University of Moratuwa, Sri Lanka

GATHIKA RATNAYAKA
Department of Computer Science & Engineering, University of Moratuwa, Sri Lanka

AMAL SHEHAN PERERA
Department of Computer Science & Engineering, University of Moratuwa, Sri Lanka

Since the advent of deep learning based Natural Language Processing (NLP), diverse domains of human society have benefited form automation and the resultant increment in efficiency. Law and order are, undoubtedly, crucial for the proper functioning of society; for without law there would be chaos, failing to offer equality to everyone. The legal domain being such a vital field, the incorporation of NLP into its workings has drawn attention in many research studies. This study attempts to leverage NLP into the task of extracting legal parties from legal opinion text documents. This task is of high importance given the significance of existing legal cases on contemporary cases under the legal practice, *case law*. This study proposes a novel deep learning methodology which can be effectively used to resolve the problem of identifying legal party members in legal documents. We present two models here, where the first is a BRNN model to detect whether an entity is a legal party or not, and a second, a modification of the same neural network, to classify the thus identified entities into petitioner and defendant classes. Furthermore, in this study, we introduce a novel data set which is annotated with legal party information by an expert in the legal domain. With the use of the said dataset, we have trained and evaluated our models where the experiments carried out support satisfactory performance of our solution. The deep learning model we hereby propose, provides a benchmark for the legal party identification task on this data set. Evaluations for the solution presented in the paper show that our system has 90.89% precision and 91.69% recall for legal party extraction from an unseen paragraph from a legal document. As for the classification of petitioners and defendants, we show that GRU-512 obtains the highest F1 score.

Keywords: Legal party identification, Recurrent Neural Networks, coreference resolution, NER

1. Introduction

Law and order are, undoubtedly, crucial for the proper functioning of society; for without law there would be chaos, failing to offer equality to everyone. The legal domain being such a vital field, the incorporation of artificial intelligence into its workings has drawn attention in many research studies. This study is also one such endeavor taken towards building an automated legal system which eventually proved to be capable of extracting information from court cases and providing analysis and

insights. The necessity of an automated legal system can be elaborated with a few major arguments, the first being the existence of case law. Case law by definition is the collection of previous judicial decisions that can be brought forth to clear the ambiguities in current cases. Therefore, the legal officials involved with a certain case are required to be knowledgeable about similar cases that have taken place before the ongoing one in hand. This task is made tedious by the abundance of documents and records in the legal domain and the unavailability of a mechanism to perform intelligent queries on the said records [34]. With a setting as such, it is apparent, that an information extraction system for legal documents can be quite beneficial since it would help the legal officials in the course of gathering relevant information. In the process of extracting information, we observed that the identification of legal parties in a case is crucial since a legal document is usually structured around the said parties where the arguments and counter-arguments are presented concerning them. For this study, we have considered using legal opinion documents in terms of the data set creation and testing. An opinion document is a statement written explaining the prevailing conditions of a case, which also contains background information at times where it is felt as required. By looking at an opinion document, one can get a sufficient understanding of the case and also of the factors leading to the decision of the case. Therefore, we believe opinion documents suits well in the attempt we take to reach a system that intelligently extracts party information.

A legal party is defined as, any individual, a group of individuals or a body that can be held accountable (i.e., organization) for law and they are usually affected by or have an interest in the outcome of the legal case [7]. Two major parties are identified in a legal case as the *petitioner*, the entity filing the case, and the *defendant*, the entity who is being prosecuted. Nevertheless, there are many complications when identifying legal parties in a document. As pointed out by Krass [17], some of the challenges faced by NLP in legal context are:

1. The unique hierarchical structure of the outcome.
2. The linguistic quirk of legal adversarial
3. Having to use acontextually trained embeddings

On top of these, we identify, *the ambiguity in deciding whether an entity should belong to a legal party or not*, also as an added hindrance to achieve our goal.

Example 1

- Sentence 1.1: *Petitioner Jae Lee moved to the United States from South Korea with his parents when he was 13.*
- Sentence 1.2: *During the plea process, Lee repeatedly asked his attorney whether he would face deportation; his attorney assured him that he would not be deported as a result of pleading guilty.*

Example 1 demonstrates the ambiguity of identifying the legal parties in a legal document. *his attorney* in Sentence 1.2 is identified to be belonging to a legal party in this case of *Jae Lee v. United States* [35] but *his parents* in Sentence 1.1 are not. This can be understood, even by a human, only after going through the entire document (or a sufficient portion of it) and comprehensively grasping the nature of the situation. An intelligent way of processing the text is therefore needed and it is obvious that there is more intrinsic work that goes into identifying legal parties than the mere identification of people (or entities) from a text.

Furthermore, extracting key legal entities from the body content of the document comes with many complexities primarily due to the unstructured nature of legal texts. For an instance, in these documents, a certain party can be referred to in various forms. The very same case of Jae Lee vs. United States [35] can be used to further explain this idea where *Jae Lee* is the petitioner. However, throughout the document, he has been referred to in different ways such as *Jae Lee*, *Lee*, *Petitioner Jae Lee*, *He*, *Lee's*, *His*, *Him*. Similarly, *United States*, which is the Respondent in the case, is referred to in various ways such as *United States* and *the Government*. Another similar concern raised in the process is the existence of several other entities (excepted from the *petitioner* and *defendant* parties) that play a significant role in the case. In legal information extraction, it is important to identify toward which party are their arguments are supportive. For an example, in the Jae Lee vs. United States [35], Jae Lee files a petition for a previous court ruling claiming he was prejudiced by his attorney (in the previous court case) and therefore *Jae Lee*, and *His Attorney* from the previous court case are opposing each other in this court case. As a result, the evidence, arguments, and opinions which are favorable for his attorney are, for the most part, favorable for *The Government* (which is the *Respondent*) as well. Conversely, the same are disadvantageous for *Lee* (*Petitioner*). This raises concerns since automatically identifying entities such as *Jae Lee's Attorney from the previous court case* and determining for which side their arguments are favorable, are two of the most important issues when it comes to party-based sentiment analysis in legal opinion texts.

Example 2

- The decision whether to plead guilty also involves assessing the respective consequences of a conviction after trial and by plea. *See INS v. St. Cyr, 533 U.S. 289,322-323*. When those consequences are, from the defendant's perspective, similarly dire, even the smallest chance of success at trial may look attractive. For Lee, deportation after some time in prison was not meaningfully different from deportation after somewhat less time; he says he accordingly would have rejected any plea leading to deportation in favor of throwing a "Hail Mary" at trial.

The paragraph extracted from Lee v. United States [35] in Example 2 is a fine demonstration of how complex the language in legal documents can get. Even though the text quote is taken without a leading text, and therefore does not make much sense out of context, it is clear that the language used in the text is relatively complex with long sentences where the writer tries to fit multiple ideas within a single sentence. Even a human who reads these documents without prior exposure to these documents may have a hard time in grasping the context since these legal documents maintain a native style of writing. Hence, there is a need for a system that has Natural Language Processing (NLP) capabilities fine-tuned for the legal domain [14, 33].

2. Related Work

The following sections briefly describe previous studies and tools which either provide the background context to our study or are directly utilized in the methodology (Section) in deriving our solution.

2.1. *NLP in the Legal Domain*

Law and order is a domain that has drawn constant attention from the research directions of NLP. Specific traits in the language used in this field have challenged the researchers and developers working on NLP tools to tackle the existing defects in those tools and has constantly pushed them to significantly

improve their work. Ontology population [14–16], discourse [26–28], semantic similarity [33, 34], and sentiment analysis [11, 22–25, 29] are some areas where extensive work has been carried out concerning the legal domain.

The Study by Gupta et al. [12] is quite similar to our study where they have worked on resolving coreference for entities that belong to a given party in legal texts. They have tried to address the issue of terms such as *petitioner*, *defendant*, *appellant* not getting included in the coreference resolution by the already existing tools developed for general purpose, which primarily only resolve coreference on the basis of pronouns.

2.2. *Coreference Resolution*

Identifying mentions of the same entity in different forms and different positions in a text can be defined as coreference resolution in simple terms. The two sentences in Example 2 are taken out from the *Lee v. United States* [35] where the two sentences appear consecutively. The words *Lee*, *his*, *he* in Sentence 2.1 and the words *His*, *him*, **he** in Sentence 2.2 refer to *Jae Lee*, who is the petitioner in this case; and the term *his attorney* in both of the sentences is a different entity who appears as a defendant in this case. A mapping between these words, which are referred to as tokens in NLP, is required to identify them as the same entity. Coreference resolution delivers to that task.

Example 3

- Sentence 3.1: *During the plea process, Lee repeatedly asked his attorney whether he would face deportation.*
- Sentence 3.2: *His attorney assured him that he would not be deported as a result of pleading guilty.*

Stanford Core[a] and *spacy*[b] are two widely available and popular tools that offer coreference resolution, both of which are built upon the work of Clark and Manning [6]. Following the initial evaluation conducted by Samarawickrama et al. [30] on both the systems for cases taken from the domain we work on, we decided to proceed with the Stanford system due to the better performance it showed. In Fig. 1 we show the result of coreference resolution conducted on the sentences in Example 2 by the online Stanford coreference resolution tool.

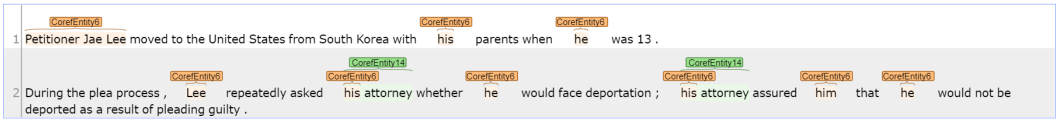

Fig. 1. Stanford Coreference Resolution of Sentences in Example 2

2.3. *Named Entity Recognition*

The task of segregating entities into predefined categories is simply known as Named Entity Recognition (NER). These categories could be anything defined by the user, but in generic NLP tools that offer NER, the available common categories include classes such as `PERSON`, `ORGANIZATION`, and

[a] https://stanfordnlp.github.io/CoreNLP/
[b] https://spacy.io/

LOCATION. The spaCy system [13] and Stanford NER system [10] can be named as the two most popular tools for NER. Similar to how we evaluated utility of the systems available for coreference resolution in Section , we evaluated these two systems with our cases and observed a better performance in the Stanford system. Therefore, we have used the Stanford NER system for the purpose of achieving NER in this study. In Fig. 2, we show the result of NER conducted on the sentences in Example 2 by the online Stanford coreference resolution tool. Note how *attorney* is tagged as TITLE instead of PERSON.

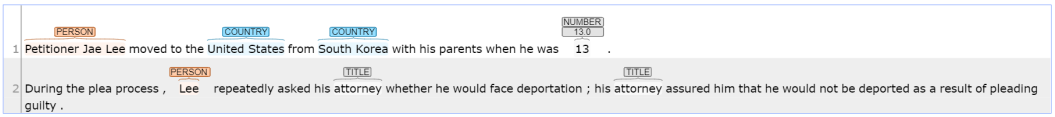

Fig. 2. Stanford Named Entity Recognition of Sentences in Example 2

2.4. *Legal Entity Identification*

Samarawickrama et al. [30] have conducted a study on the same research problem handled in this study, albeit a static rule-based approach to achieve legal entity identification is presented. In their work, coreference resolution is first performed and clusters of entities are identified in the document. Then, the identified clusters are filtered out where only the entities which are either a person or an organization are retained. Afterward, the number of times in which each entity appears as subjects in the text is taken into consideration when calculating the probability of each entity being an actual legal entity.

2.5. *Sequence-to-Sequence Learning for Legal Party Identification*

This is a similar attempt to party identification in legal documents to that of de Almeida et al. [7], given that it too is conducted using coreference resolution and named entity recognition. However, it differs from de Almeida et al. [7] by the fact that it utilizes deep learning as its approach to identify the legal parties. A Recurrent Neural Network (RNN) encoder-decoder model with Long Short Term Memory (LSTM) cells is used in this study, where masked sentences (a mask is applied to person/organization entities) are used as the input. The model gives an output sequence of 1s and 0s where, a 1 denotes that the corresponding token is referring a legal party in the document and a 0 denotes otherwise. This model uses character-wise encoding to represent the words, which means, it assigns a real value to each unique character to form the word. In this study, we have further improved this approach by introducing word embeddings to replace simple character-wise encoding, since we observed that the vectors would help to carry more information content with the higher dimensionality it offers.

2.6. *Recurrent Neural Networks*

Recurrent Neural Network (RNN) is a class of artificial neural networks that are best suited for dealing with sequential data. RNNs ability to process inputs of variable length has helped it to show excellent performance in NLP related tasks [2]. Gated Recurrent Unit(GRU) [5] or Long Term Short Term Memory (LSTM) [32] cells can be used to retain the previous hidden state and the current input in RNNs. This study experiments with both of these approaches to find the best-suited architecture for its

intended task. The obtained results for this are presented in Section where we discuss the experiments.

2.7. *Bidirectional Recurrent Neural Networks*

Future input information is also important for prediction. This can be solved with RNN by delaying the output by a certain number of time steps to use future information for current prediction. But all future information cannot be captured. Bidirectional Recurrent Neural Network (BRNN) that can be trained using all available input information in the past and future of a specific time frame has been proposed by Schuster and Paliwal [31] to overcome the limitations of a regular RNN. The state neurons of a regular RNN are divided into two parts. One is responsible for the positive time direction and the other is for the negative time direction. Outputs from positive time direction and inputs of negative time direction are not connected, and vice versa. So that the BRNN can be trained with the same algorithms as a normal RNN.

2.8. *Word Embedding*

Word embedding is a technique used to represent words in the form of vectors while preserving their meaning. As well as other domains, many research studies have been conducted in the field of law and order with the incorporation of word embedding due to the fine performance this technology offers. Jayawardana et al. [15] used word embeddings to derive a representative vector for classes in a legal ontology. Then they used the same word embeddings to help with the automatic population of the legal ontology [14, 16]. Sugathadasa et al. [33] created a domain specific similarity measure for the legal domain using word embedding. They then used the said word embedding and document embedding to build a legal document retrieval system [34].

In this study, similar to Sugathadasa et al. [34], we have incorporated a pre-trained Word2Vec [19–21] model to create vector embedding of the tokens in the text we use to train our extended model. We use pre-trained vectors trained on part of the Google News data set where each word is represented by a vector of dimension 300[c].

3. Methodology

In this study we propose a novel method to accurately identify the members of the legal parties involved in a legal case using a legal opinion paragraph (hitherto referred as *paragraph*). Also note that from here onward, the members of the legal parties involved in the case are referred to as *party members*. Our method consists of four main steps: *Tokenizing* (Discussed in Section), *Embedding* (Discussed in Section), *Masking* (Discussed in Section), and *Neural Network Model* (Discussed in Section). The full flow of the proposed methodology is shown in Figure 3.

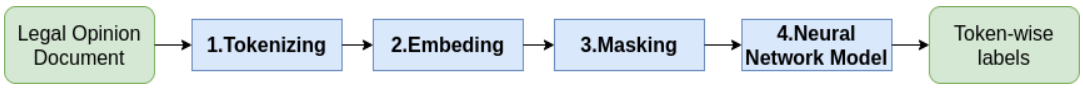

Fig. 3. Methodology Process Flow

3.1. *Tokenizing*

[c] https://code.google.com/archive/p/word2vec/

First, the given paragraph is passed onto the Stanford annotator to split the text into a sequence of tokens and to get the Coreference Resolution and NER results of each token. Tokens can be either words, numbers, or punctuation marks. Out of these tokens, the entities that are either a `PERSON`, an `ORGANIZATION`, or a `LOCATION` are identified with the use of named entity recognition because only those entities can be a *party member*. Afterward, coreference resolution is performed on thus identified `PERSON` / `ORGANIZATION` / `LOCATION` entities, and their corresponding mentions are identified to form groupings of the tokens, to which in this paper we refer to as, *coref clusters*. The NER value of the headword of each *coref cluster* is then passed down to the other tokens of the *coref cluster* and a mapping is created to store the information token-wise (the fact that a certain token refers back to a `PERSON` / `ORGANIZATION` / `LOCATION` entity) to generate the masks later in Section .

3.2. *Embedding*

We decided to use word2vec [19–21] as the word embedding method because it generates vectors for a given word without considering the frequency of co-occurrences. In this study, we train the model with *paragraphs* of different sizes. Therefore, the frequency of a word is not important. In this step, a vector, that is of dimension 300, for each token is generated with the use of a pre-trained model created by Google.[d] Before generating the vector, we also make sure to pass the token through a stemmer to increase the probability of the model containing the word. In a scenario where the model does not have an already trained vector for the token (This may happen for tokens such as: numbers, punctuation marks, proper nouns), a zero vector with the same dimension is returned as the corresponding vector for the token. We decided to use word2vec because we needed to get word-wise embedding by considering each word as an atomic entity and the study by Sugathadasa et al. [33] and the study by Jayawardana et al. [14] recommended it for the use in the legal domain.

3.3. *Masking*

In this step, an additional value (v) is generated for tokens that are identified as either a `PERSON` or an `ORGANIZATION`, or a `LOCATION`. The logic that goes into deciding this value is explained with the Algorithm 1. The mask value (v) takes a value between 0 and 1 where the range is further divided into smaller ranges for `PERSON`, `ORGANIZATION`, and `LOCATION` classes. A `PERSON` entity gets a mask value v of range $0 < v < 0.5$, an `ORGANIZATION` entity gets a mask value v of range $0.5 < v < 0.75$, and a `LOCATION` entity gets a mask value v of range $0.75 < v < 1.0$. The reason we assigned a wider range to the `PERSON` entities in comparison with the other two is that we observed in opinion texts, `PERSON` entities appear in higher frequencies than the other two. All the tokens of the same *coref cluster* identified in the tokenizing step discussed in Section , are given the same value. The vector of each token given by the previous step are masked or extended with values generated for each token by Algorithm 1.

Example 4

- The Federal Trade Commission (FTC) filed an administrative complaint, alleging that the Board's concerted action to exclude non dentists from the market for teeth whitening services in North Carolina constituted an anti competitive and unfair method of competition under the Federal Trade Commission Act.

[d] https://code.google.com/archive/p/word2vec/

In the Example 3, *The Federal Trade Commission (FTC)* and *the Board* are `ORGANIZATION` entities. On the other hand, *North Carolina* is a `LOCATION` entity. Thus, the tokens: *The*, *Federal*, *Trade*, *Commission*, *(*, *FTC*, and *)* are masked with a value v_1 such that $(0.5 < v_1 < 0.75)$, the tokens: *the* and *Board* are masked with a value v_2 such that $(0.5 < v_2 < 0.75)$, and the tokens: *North* and *Carolina* are masked with a value v_3 such that $(0.75 < v_3 < 1.0)$.

Algorithm 1 Mask Value Generator

Input: $corefs, NERvaluesofheadwordsofclusters$
Output: $D_{maskValues}$

procedure VALGENERATOR($corefs$)
  for `each` $cluster_k$ `in corefs` **do**
    $head_k \longleftarrow cluster_k.headword$
    if $head_k.NER ==$ `PERSON` **then**
      $val_k \longleftarrow v_1(0 < v_1 < 0.5)$
    else if $head_k.NER ==$ `ORGANIZATION` **then**
      $val_k \longleftarrow v_2(0.5 < v_2 < 7.5)$
    else if $head_k.NER ==$ `LOCATION` **then**
      $val_k \longleftarrow v_3(0.75 < v_3 < 1.0)$
    else
      $val_k \longleftarrow 0$
    end if
  end for
  $D_{maskValues} \longleftarrow setOf(head_k : val_k)$
end procedure

3.4. *Neural Network Model for party identification*

The input of this model is the sequence of masked vectors $\{X_1, ..., X_T\}$ of the given *paragraph* and the output is a sequence of probabilities $\{y_1, ..., y_T\}$, where T is the number of tokens in the *paragraph*. If a token ($token_t$) of the given *paragraph* is referred to as a *party member* then the output value (y_t) corresponding to that token should be close to 1. Otherwise it should be close to 0. Figure 4 depicts the architecture of the model we designed for this task. We feed the sequence of masked vectors $\{X_1, ..., X_T\}$ into a sequence of BRNN cells (BRNN layer). We use BRNN instead of regular RNN because the model needs all the information about the given *paragraph* to decide whether any token is referring to a *party member* or not. The sequence of output vectors generated by the BRNN layer, $\{U_1, ..., U_T\}$, is then fed into dense layers to generate the probability of each token to be referred to a *party member*. We perform a token-wise binary classification at the output layer of the model using the sigmoid activation function.

In Example 3, *The Federal Trade Commission (FTC)* and *the Board* are two *party members* involved in the case. So that the tokens: *The*, *Federal*, *Trade*, *Commission*, *(*, *FTC*, *)*, *the*, and *Board* are referred to *party members*. Therefore, outputs corresponding to those tokens should be 1. Outputs corresponding to all the other tokens should be 0.

3.4.1. *Training Phase*

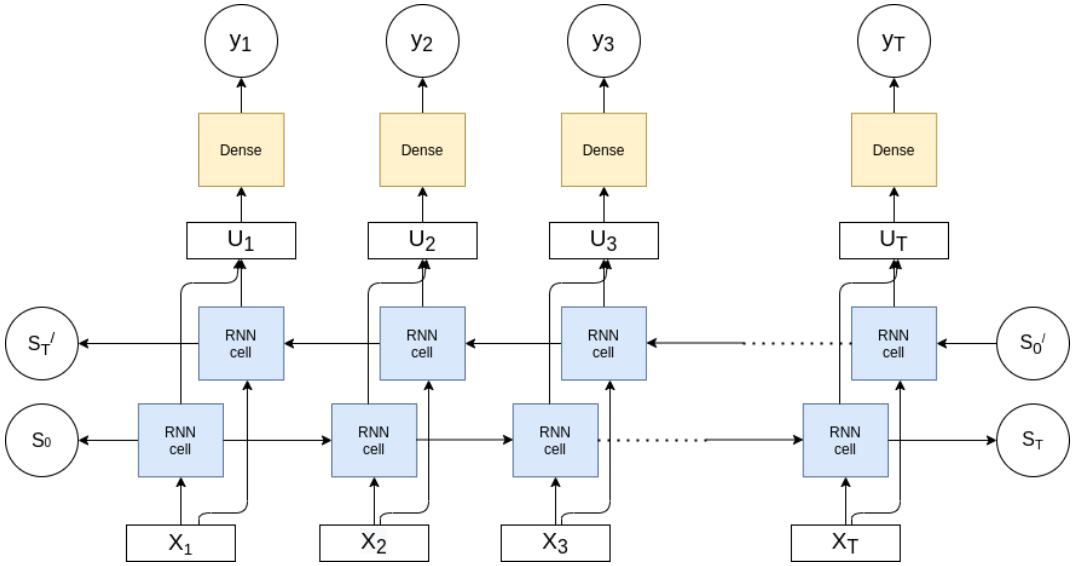

Fig. 4. Architecture of the Neural Network Model for party identification

In this phase, Neural Network Model is trained to minimize the Binary Cross-Entropy using a data set that consists of a 3D input array of size (m,n,T) and a 2D expected output array of size (m,T). 3D input array contains a set of sequences of vectors. 2D expected output array contains the corresponding set of sequence of binary values (1s and 0s). In this phase, the model essentially learns parameters of BRNN and Dense layers to identify tokens of a text sample that refer to a *party member*.

m = number of sample paragraphs

n = dimension of a token vector after masking (301)

T = maximum possible number of tokens in a given sample paragraph

If the number of tokens in a given sample paragraph is less than T, zero vectors of size n are added to fill the 3D array.

3.4.2. *Inference Phase*

In this phase, the model uses learned parameters of BRNN and Dense layers to identify tokens of a text sample that are referred to as a *party member*. To achieve this end, the trained Neural Network Model takes the masked word embedding of the given paragraph, which is an array of size (n,T), and gives the corresponding result vector of size T. Each value of the result vector is between 0.0 and 1.0, which is the probability of each token to be referred to a *legal party* of the *paragraph*. Additionally, in this phase, if the number of tokens in a given *paragraph* is less than T, zero vectors of size n are added to fill the array.

3.5. *Modified Neural Network Model for Petitioner and Defendant Identification*

This model is responsible for performing token-wise multi-label classification, given the sequence of vectors $\{v_1, ..., v_T\}$ generated by the Word2Vec model. Fundamentally, it outputs the probability of each token $\{token_1, ..., token_T\}$ being a mention of a member of the petitioner $\{prob\_p_1, ..., prob\_p_T\}$ and the probability of each token being a mention of the defendant $\{prob\_d_1, ..., prob\_d_T\}$ involved in the case. We used multi-label classification instead of multi-class classification because, there can be some neutral entities which may appear to be members of both the petitioner and the defendant in the same case. Figure 5 depicts the architecture of the Modified NN Model, we designed for this task. It consists of a sequence of T number of bidirectional Modified NN cells, followed by a Dense layer with sigmoid activation function. Here note the difference from Figure 4, as separate p_i and d_i values are predicted rather than a singular y_i value. This also has two phases: Training Phase and Test Phase.

3.5.1. *Training Phase*

In this phase, the algorithm takes the sequences of vectors (D) of all the training samples and expected outputs (L) and trains the Modified NN Model. Here, D is a 3D array of shape (m, T, n), where m is the number of training samples and n is the size of the vector of a single token ($n = 301$). L is a 3D array of shape $(m, T, 2)$.

3.5.2. *Test Phase*

In this phase, the algorithm takes the sequence of vectors $\{v_1, ..., v_T\}$ generated by the Word2Vec for the current text instance, passes it through the trained Modified NN Model and outputs the probability of each token to be a mention of the petitioner $\{prob\_p_1, ..., prob\_p_T\}$ and the probability of each token to be a mention of the defendant $\{prob\_d_1, ..., prob\_d_T\}$.

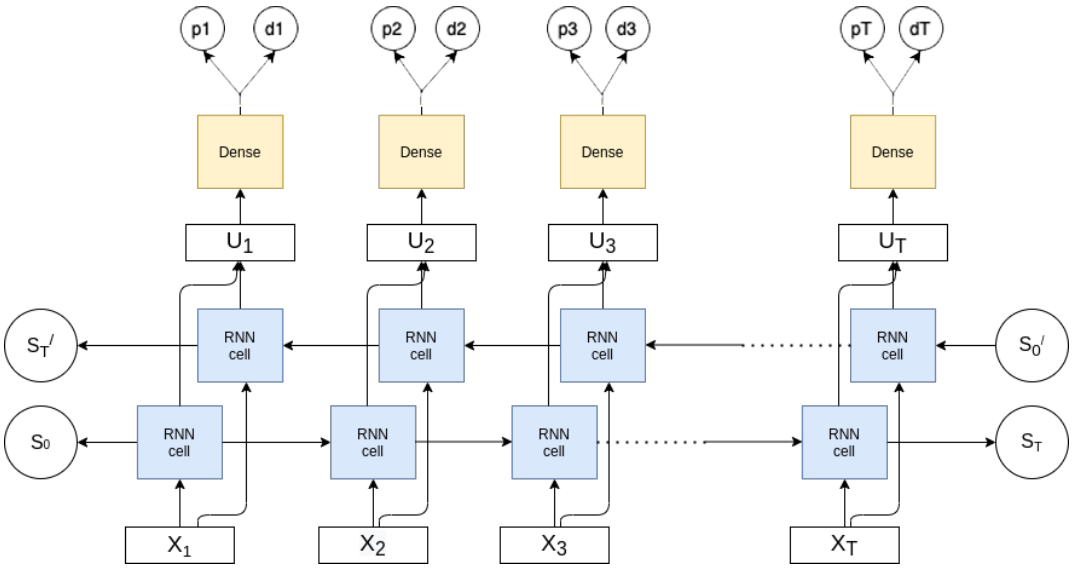

Fig. 5. Architecture of the Neural Network Model for identification for petitioner and defendant separately

4. Experiments

This section consists of the details of the experiments we conducted to evaluate the performance of our solution. Sections and respectively comprise of the details regarding the external libraries used for implementation and the details about the data set used for training. The following Section presents the results along with a brief explanation of the performance evaluation metrics used.

4.1. *Setup*

- **Natural Language Processing:** Stanford CoreNLP tools were used to perform NER and coreference resolution and for stemming purposes, the Porter stemmer by NLTK [3, 18] was also used.
- **Deep Learning :** We used Keras [4] open source library which runs on top of Tensorflow [1] python library to implement the Neural Network Model.
- **Word Embedding:** We used a pretrained Word2Vec [19–21] model by Google to generate vectors in the Embedding step.

4.2. *Data Set*

Due to the unavailability of token-wise labeled data for petitioner and defendant identification, we created and publicly shared a custom data set[e] with the help of an expert of the legal domain for training and testing the model. This data set consists of 1000 paragraphs that are picked from legal opinion documents of 1000 different legal cases written in the English Language from the 39,155 legal cases data set[f] published by Sugathadasa et al. [33]. We tokenized each *paragraph* using the tokenizing method we explained in Section . Then, we manually labeled each token of each *paragraph* as a *party member* mentioned in each sample or not with the help of an expert in the legal domain. Then we generated masked word embedding of each paragraph by following the embedding (Section) and masking (Section) steps. The statistics of our data set are mentioned in Table 1. The number of tokens that are referred to as *party members* is just about 3.46% of the total number of tokens of a *paragraph*. Hence, by itself proving the difficulty and the importance of this study by the same argument raised by de Silva [9] on the case of inconsistency in research paper abstracts.

Table 1. Statistics of the Data Set

Attribute	Train	Validation	Test	Total
Paragraphs	810	90	100	1000
Tokens	351K	39K	43K	433k
Party member tokens	12.16K	1.21K	1.62K	14.99K

4.3. *Performance of the Neural Network Model for Party Identification*

This model was trained for 100 epochs with a learning rate of 0.01 separately with Gated Recurrent Units (GRU) and Long short-term memory (LSTM) with alterations done to the number of output

[e] https://osf.io/eahg8/
[f] https://osf.io/w3paz/

units of the BRNN layer. We used Binary Cross Entropy as the loss function with Adam Optimizer. We used Precision at the given Recall as the metric to train the model because according to Table 1, the number of expected positives is much less than the number of expected negatives.

The performance of the model according to Accuracy (A), Precision (P), Recall (R), $F1$ score, and the average training time per step is shown in Table 2. The equations 1, 2, 3, and 4 are the definitions of Accuracy (A), Precision (P), Recall (R), $F1$ score, where, TP is the count of True Positives, TN is the count of True Negatives, FP is the count of False Positives, and FN is the count of False Negatives.

$$A = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

$$P = \frac{TP}{TP+FP} \tag{2}$$

$$R = \frac{TP}{TP+FN} \tag{3}$$

$$F1 = \frac{2*P*R}{P+R} \tag{4}$$

Table 2. Performance of the Neural Network Model

BRNN cell type	Output units	Performance of the model over test set				
		Accuracy (A) (%)	Precision (P) (%)	Recall (R) (%)	$F1$ score (%)	Training time per step (s)
GRU	8	97.7	70.4	44.8	54.7	0.4
	32	98.4	78.9	58.2	66.9	0.6
	64	98.9	79.8	73.1	76.3	1.0
	128	99.1	79.5	79.2	79.4	2.0
	256	99.6	86.5	86.6	86.5	4.0
	512	99.9	90.9	91.7	91.3	15.0
LSTM	8	97.6	67.8	34.6	45.8	0.4
	32	98.4	74.8	58.9	65.9	0.7
	64	98.7	81.3	63.9	71.6	2.0
	128	99.3	85.0	79.7	82.3	3.0
	256	99.5	87.5	84.3	85.9	7.0
	512	99.8	89.7	92.2	90.9	20.0

Training times shown in this table 2 are according to the performance of the Intel Xeon Processor with two cores @ 2.30 GHz and 13GB RAM. We can see that as the number of output units increases, the Accuracy (A), Precision (P), Recall (R), $F1$ score of models with both GRU and LSTM has increased. The model with GRU cells of 512 output units has shown the best performance. Also,

the time consumption of GRU is much less than LSTM when the complexity of the model increases. These results are a clear indication of the accuracy of our methodology to identify legal party members.

4.4. *Performance of the Neural Network Model for Petitioner and Defendant Identification*

We trained the modified RNN model separately with Gated Recurrent Units (GRU) and Long Long short-term memory (LSTM) with alterations done to the number of output units of the RNN layer (u), taking binary cross-entropy as the loss function and recall at precision 0.7 as the metric. We focused mainly on improving recall (R, equation 3) because, there is a tendency for the model to generate false negatives, as the number of tokens that are mentions of the petitioner or the defendant are respectively only 1.75% and 1.62% of the total number of tokens present in the legal opinion text according to the statistics of the dataset (Table 3).

Table 4 shows the performance of this model in identifying the tokens that are mentions of the *petitioner* and the tokens that are mentions of the *defendant*, for GRU and LSTM of different u. Accuracy (A), precision (P), recall(R), $F1$ score and Training time per step in seconds (s) are used as performance measures, taking 0.5 as the baseline accuracy given that this, in-essence, is a binary classification problem. Training times showed in these tables are according to the performance of Intel Xeon Processor with two cores @ 2.30 GHz and 13GB RAM. We trained this model for 100 epochs with Adam optimization with learning rate of 0.001. RNN Model with GRU cells of 512 output units (GRU-512) has the highest of $F1$ score for identifying the mentions of the *petitioners* and the *defendants*. Therefore we decided to proceed with the experiments of the entire methodology using these two trained RNN Models.

Table 3: Statistics of the Dataset

Attribute	Train	Val	Test	Fullset
Cases	630	70	50	750
Tokens	279K	31K	22K	332K
Petitioner tokens	4820	550	424	5794
Defendant tokens	4391	577	408	5376

5. Conclusion

In this study, which is an extension of our conference paper [8], we propose a natural language processing method to accurately predict the members of *legal parties*, given a *paragraph* of a legal opinion document. First, We identify the entities that are either a `PERSON` or an `ORGANIZATION` or a `LOCATION` and spot their mentions in the paragraphs. Our model then proceeds to evaluate the likelihood of each such mention to be referred to as a legal party member by inspecting the meaning of the paragraph using BRNN. The meaning is grasped by the model with the help of word embedding and learned through the training process. Also, we introduce a data set that can be used for future research. We show that our system has 90.89% precision and 91.69% recall for an unseen *paragraph* from a legal document. Therefore, this method can be used to identify the entities that are most likely to be a *legal*

Table 4: Performance of the Modified NN Model

BRNN cell type	Output units	Petitioner				Defendant				Training time per step (s)
		A(%)	P(%)	R(%)	F1(%)	A(%)	P(%)	R(%)	F1(%)	
GRU	8	99.1	71.9	54.6	60.4	99.3	53.8	44.8	46.9	0.1
	32	99.8	82.0	82.5	82.1	99.8	70.3	69.2	69.4	0.2
	64	99.9	88.5	88.1	88.2	99.9	73.0	73.0	73.0	0.3
	128	99.9	88.4	88.8	88.5	99.9	73.8	73.8	73.8	0.6
	256	99.9	88.7	88.2	88.4	99.9	73.2	74.2	73.6	2.0
	512	99.9	89.0	89.0	89.0	99.9	75.0	75.0	75.0	5.0
LSTM	8	99.0	70.9	50.0	56.6	99.2	50.2	39.3	42.0	0.1
	32	99.7	78.8	77.8	77.7	99.7	66.6	65.0	65.3	0.2
	64	99.8	87.7	84.3	85.7	99.9	72.2	70.8	71.1	0.4
	128	99.9	87.1	87.3	87.1	99.9	73.9	74.7	74.3	0.7
	256	99.9	89.0	88.9	88.9	99.9	74.6	74.6	74.6	2.0

party.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[3] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[4] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[6] Kevin Clark and Christopher D Manning. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*, 2016.

[7] Melonie de Almeida, Chamodi Samarawickrama, Nisansa de Silva, Gathika Ratnayaka, and Amal Shehan Perera. Legal Party Extraction from Legal Opinion Text with Sequence to Sequence Learning. In *2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 143–148. IEEE, 2020. doi: 10.1109/ICTer51097.2020.9325488.

[8] Melonie de Almeida, Chamodi Samarawickrama, Nisansa de Silva, Gathika Ratnayaka, and Shehan Perera. Identifying Legal party Members from Legal Opinion Text using Natural Language Processing. In *The 23rd International Conference on Information Integration and Web Intelligence*, pages 259–266, 2021. doi: 10.1145/3487664.3487700.

[9] Naida Hewa Nisansa Dilushan de Silva. *Semantic Oppositeness for Inconsistency and Disagreement Detection in Natural Language*. PhD thesis, University of Oregon, 2020. URL https://search.proquest.com/openview/3def92003aaab0901d896bf3919034ba/1?pq-origsite=gscholar&cbl=18750&diss=y.

[10] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, pages 363–370, 2005.

[11] Viraj Gamage, Menuka Warushavithana, Nisansa de Silva, Amal Shehan Perera, Gathika Ratnayaka, and Thejan Rupasinghe. Fast Approach to Build an Automatic Sentiment Annotator for Legal Domain using Transfer Learning. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 260–265, 2018. doi: 10.18653/v1/W18-6238.

[12] Ajay Gupta, Devendra Verma, Sachin Pawar, Sangameshwar Patil, Swapnil Hingmire, Girish K Palshikar, and Pushpak Bhattacharyya. Identifying participant mentions and resolving their coreferences in legal court judgements. In *International Conference on Text, Speech, and Dialogue*, pages 153–162. Springer, 2018.

[13] Matthew Honnibal and Ines Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 7(1), 2017.

[14] V. Jayawardana, D. Lakmal, Nisansa de Silva, A. S. Perera, K. Sugathadasa, B. Ayesha, and M. Perera. Word Vector Embeddings and Domain Specific Semantic based Semi-Supervised Ontology Instance Population. *International Journal on Advances in ICT for Emerging Regions*, 10(1):1, 2017. doi: 10.4038/icter.v11i1.7191.

[15] Vindula Jayawardana, Dimuthu Lakmal, Nisansa de Silva, Amal Shehan Perera, Keet Sugathadasa, and Buddhi Ayesha. Deriving a Representative Vector for Ontology Classes with Instance Word Vector Embeddings. In *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*, pages 79–84. IEEE, 2017. doi: 10.1109/intech.2017.8102426.

[16] Vindula Jayawardana, Dimuthu Lakmal, Nisansa de Silva, Amal Shehan Perera, Keet Sugathadasa, Buddhi Ayesha, and Madhavi Perera. Semi-Supervised Instance Population of an Ontology using Word Vector Embedding. In *Advances in ICT for Emerging Regions (ICTer),*

2017 Seventeenth International Conference on, pages 1–7. IEEE, September 2017. doi: 10.1109/icter.2017.8257822.

[17] M. Krass. Learning the rulebook: Challenges facing nlp in legal contexts. 2019.

[18] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[21] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[22] Chanika Ruchini Mudalige, Dilini Karunarathna, Isanka Rajapaksha, Nisansa de Silva, Gathika Ratnayaka, Amal Shehan Perera, and Ramesh Pathirana. Sigmalaw-absa: Dataset for aspect-based sentiment analysis in legal opinion texts. *arXiv preprint arXiv:2011.06326*, 2020. doi: 10.1109/ICIIS51140.2020.9342650.

[23] Isanka Rajapaksha, Chanika Ruchini Mudalige, Dilini Karunarathna, Nisansa de Silva, Gathika Rathnayaka, and Amal Shehan Perera. Rule-based approach for party-based sentiment analysis in legal opinion texts. *arXiv preprint arXiv:2011.05675*, 2020. doi: 10.1109/ICTer51097.2020.9325435.

[24] Isanka Rajapaksha, Chanika Ruchini Mudalige, Dilini Karunarathna, Nisansa de Silva, Amal Shehan Perera, and Gathika Ratnayaka. Sigmalaw PBSA-A Deep Learning Model for Aspect-Based Sentiment Analysis for the Legal Domain. In *International Conference on Database and Expert Systems Applications*, pages 125–137. Springer, 2021. doi: 10.1007/978-3-030-86472-9_12.

[25] Isanka Rajapaksha, Chanika Ruchini Mudalige, Dilini Karunarathna, Nisansa de Silva, Gathika Ratnayaka, and Amal Shehan Perera. Sigmalaw PBSA-A Deep Learning Approach for Aspect-Based Sentiment Analysis in Legal Opinion Texts. *Journal of Data Intelligence*, 3(1):101–115, 2021. doi: https://doi.org/10.26421/JDI3.1-1.

[26] G. Ratnayaka, T. Rupasinghe, Nisansa de Silva, M. Warushavithana, V. Gamage, M. Perera, and A. S. Perera. Classifying Sentences in Court Case Transcripts using Discourse and Argumentative Properties. *ICTer*, 12(1), 2019. doi: 10.4038/icter.v12i1.7200.

[27] Gathika Ratnayaka, Thejan Rupasinghe, Nisansa de Silva, Menuka Warushavithana, Viraj Gamage, and Amal Shehan Perera. Identifying Relationships Among Sentences in Court Case Transcripts Using Discourse Relations. In *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 13–20. IEEE, 2018. doi: 10.1109/ICTER.2018.8615485.

[28] Gathika Ratnayaka, Thejan Rupasinghe, Nisansa de Silva, Viraj Gamage, Menuka Warushavithana, and Amal Shehan Perera. Shift-of-Perspective Identification Within Legal Cases. In *Proceedings of the 3rd Workshop on Automated Detection, Extraction and Analysis of Semantic Information in Legal Texts*, 2019.

[29] Gathika Ratnayaka, Nisansa de Silva, Amal Shehan Perera, and Ramesh Pathirana. Effective approach to develop a sentiment annotator for legal domain in a low resource setting. *arXiv preprint arXiv:2011.00318*, 2020.

[30] Chamodi Samarawickrama, Melonie de Almeida, Nisansa de Silva, Gathika Ratnayaka, and Amal Shehan Perera. Party Identification of Legal Documents using Co-reference Resolution and Named Entity Recognition. In *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)*, pages 494–499. IEEE, 2020. doi: 10.1109/ICIIS51140.2020.9342720.

[31] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[32] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[33] Keet Sugathadasa, Buddhi Ayesha, Nisansa de Silva, Amal Shehan Perera, Vindula Jayawardana, Dimuthu Lakmal, and Madhavi Perera. Synergistic Union of Word2Vec and Lexicon for Domain Specific Semantic Similarity. *IEEE International Conference on Industrial and Information Systems (ICIIS)*, pages 1–6, 2017. doi: 10.1109/ICIINFS.2017.8300343.

[34] Keet Sugathadasa, Buddhi Ayesha, Nisansa de Silva, Amal Shehan Perera, Vindula Jayawardana, Dimuthu Lakmal, and Madhavi Perera. Legal Document Retrieval using Document Vector Embeddings and Deep Learning. In *Science and Information Conference*, pages 160–175. Springer, 2018. doi: 10.1007/978-3-030-01177-2_12.

[35] Supreme Court. Lee v. United States. *US*, 432(No. 76-5187):23, 1977.