University of Moratuwa

FINAL YEAR PROJECT

FINAL REPORT

Party Identification of Legal Documents

Authors
Melonie de Almeida
(160097G)
Chamodi Samarawickrama
(160548R)

Supervisors
Dr. Amal Shehan Perera
Dr. Nisansa de Silva
Mr. Gathika Ratnayaka

February 12, 2021

Contents

Chapter 1

Introduction

Law and order is the core guideline in a society that is predominantly responsible for maintaining the peace and conduct among citizens. It in essence keeps society running for without law there would be chaos and it would be survival of the fittest and every man for himself. Not an ideal lifestyle for most part [1]. Therefore, it is obvious that any advancement made to the field of law is in fact an advancement done towards the betterment of the society. This research is an effort made to contribute to a legal system that would be well capable of extracting information from court cases and analyzing them, providing the users with easy and meaningful access to the information residing in them. This research particularly looks at identifying the legal entities involved in a given legal case.

Case law is a form of law that is created by the courts in the process of passing judgement for a case. With the existence of case law, interpretations of the laws done in previously decided cases can be used as precedents to support arguments in an ongoing trial. Therefore, it is expected of the lawyers and other legal officials to be knowledgeable on cases that are similar to the one in the hearing. With a setting as such, arises the necessity of an automated system that can extract information out of legal documents with a Natural Language Understanding(NLU) approach. The tediousness associated with browsing through the excessive amount of documents against the limited time available to examine them further motivates for such a system. A legal opinion is a form of legal documentation that can immensely aid as a resource in building such a system since these documents contain legal conclusions and analysis done by lawyers regarding a legal transaction. And when observing opinion texts, it can be seen that the arguments and the counter-arguments presented there often stem out from the parties involved in the case. Therefore, we believe that legal Party extraction stands out with high significance in the process of extracting meaningful information out of legal opinion texts. Nevertheless, we face many challenges in the process of advocating Natural Language Processing(NLP) into working with legal documents. Learning the Rulebook: Challenges Facing NLP in the Legal Contexts [2] points out three roadblocks to incorporate NLP in legal context, as: 1) the unique hierarchical structure of outcomes, 2) the linguistic quirk of legal adversarial and 3) the challenge of using acontextually trained embeddings.

In addition to these more generic issues found with using NLP for the legal domain, the ambiguous nature of the legal parties also makes the party extraction complicated. That is the fact that a mere person's appearance in the text does not qualify the said entity to be a legal party; for example, a witness in a case is not considered to be a party involved in that case. Additionally, in opinion texts title terms like *petitioner, defendant* appear before an entity name. Nevertheless, it cannot be identified with full confidence that an entity having the said title belongs to the same party since they could have acted according to the title in a previous case but not in the case at discussion.

Example 1

- This appeal primarily concerns the proper application of the presumption of fault, applicable in maritime law, known as the ”Oregon rule.” See The Oregon, 158 U.S. 186, 15 S.Ct. 804, 39 L.Ed. 943 (1895). Defendant-Appellant Hornbeck Offshore Transportation, LLC (”Hornbeck”) appeals from the January 8, 2008, judgement of the District Court(Kevin N. Fox, MagistrateJudge), finding Hornbeck liable to Plaintiffs-Appellees Zerega Avenue Realty Corp.(”Zerega”) and Fred Todino & Sons, Inc...

The opinion text extraction presented in the Example 1 demonstrates the above explained complication where Defendant-Appellant Hornbeck Offshore Transportation, LLC, eventhough having the title Defendant, acts as a petitioner in this case while Plaintiffs-Appellees Zerega acts as a defendant.

The Example 1 is a paragraph extracted from *Lee v. United States* [3] to demonstrate the complexity of the language used in legal documents. Even though the above text quote is taken without a leading text and therefore does not make much sense in the 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

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. Pointing to Strickland, the Government urges that"[a] defendant has no entitlement to the luck of a lawless decision maker." 466 U. S., at 695. That statement, however, was made in the context of discussing the presumption of reliability applied to judicial proceedings, which has no place where, as here, a defendant was deprived of a proceeding altogether.

that has NLP capabilities fine tuned for the legal domain.

A party in a legal context stands for a single person or a group of persons that can be identified as entities to bear accountability for the purpose of law. Majorly, it can be a participant in a lawsuit or any other legal proceeding, the outcome of which will interest the said participant. Two parties are often found involved with a case; namely the plaintiff; that is whoever the legal entity filing the suit, and a defendant, the entity sued or charged against. Therefore, a party may or may not be an actual person but rather a certain body that has decisive capabilities. Additionally, a person who only appears in the case to provide witnesses is not considered to be a party. Thus, it is clear that the process of identifying the parties in a legal case has to do a lot more than identifying mere individuals or entities appearing to be similar figures. This research is an attempt made in automating a successful means to correctly identify the entities that belong to a party in a legal case.

Example 2 contains two sentences extracted from *Lee v. United States* [3] which demonstrates the ambiguity in deciding legal entities in a legal document. It turns out that the *his parents* in Sentence 1.1 does not qualify to be a legal entity in this case but is a mere mention of them to explain the history of *Petitioner Jae Lee*. On the contrary, a similar case presented as the Sentence 1.2 contains one of the major legal entities in this case which here appears as *his attorney*. By carefully analysing the entire document, we can reach that understanding of information but

Example 3

- Sentence 3.1: *Petitioner Jae Lee moved to the United States from South Korea with his parents when he was 13.*
- Sentence 3.2: *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.*

that is by running the text through various sieves in our brain and making logical assumptions and whatnot. With this paper, we propose a rudimentary model to identify the legal entities in the document. Party extraction from legal documents is crucial for the extraction of information from unstructured legal documents and analysing them, since the arguments in the said documents are presented with relation to the parties at hand. This study proposes a methodology to extract the petitioners and the defendants in a legal opinion text

Chapter 2

Problem Statement

Party extraction from legal documents is crucial for the extraction of information from unstructured legal documents and analysing them, since the arguments in the said documents are presented with relation to the parties at hand. This study proposes a methodology to extract the petitioners and the defendants in a legal opinion text.

Chapter 3

Research Objectives

- Introducing a useful dataset regarding legal parties in legal documents
- Designing preprocessing methods that would best suit the task of extracting legal parties from legal documents
- Incorporating deep learning methods into legal party extraction
- Designing a best fitting encoding method for the deep learning model with relevance to legal party extraction
- Designing an effective application of the existing NLP tools in legal party extraction
- Contributing to an intelligent system that is capable of analyzing legal documents

Chapter 4

Literature Review

4.1 NLP in Legal Domain

Law and order is a domain that has drawn constant attention in the research areas of Natural Language Processing. Specific traits in the language used in this field have challenged the researchers and developers working on NLP tools to tackle the already existing defects in them and has constantly pushed them to level up their work. Ontology population[4, 5], discourse[6, 7] and semantic similarity[8, 9] are some areas where extensive work has been carried out concerning the legal domain. Krass in his study Learning the Rulebook: Challenges Facing NLP in Legal Contexts [2] identifies three major hindrances that come along in incorporating NLP into the legal domain. He trains 4 CNNs on a dataset of 300,000 decisions by the Board of Vetaran's Appeals(BVA) to predict whether a decision by the BVA will be a) not appealed or b) appealed and if appealed, whether it will be i) affirmed, ii) reversed/remanded or iii) dismissed. In the process of achieving this task, he identifies three roadblocks to use NLP in a legal context which are the unique hierarchical structure of outcomes, the linguistic quirk of legal adversarialism and the challenge of using contextually trained embeddings. We can identify the work carried out in relation to NLP in legal domain in three categories that are the legal data search(retrieving and classifying relevant legal data texts), legal text analytics(summarization, text prediction, identifying sections in legal documents, translation, element extraction from documents) and legal intelligent interfaces(question answering bots, judgment prediction systems). Out of these our research falls into the text analytic category where we make an effort to extract the legal parties which is a form of element extraction. Chalkidis et al. [10]

in their study use deep learning to extract contract elements where they incorporate word, POS tag and tokenshape embeddings to implement a Bi-LSTM model. With this model they have managed to extract certain important elements from contracts such as termination dates, legislation references, contracting parties and agreed payments.

Identifying Participant Mentions and Resolving Their Coreferences in Legal Court Judgements [11] is a work that is quite similar to our study where Gupta et al. have worked on resolving coreference for entities that belong to a party in legal texts. They have tried to address the issue of terms like *petitioner*, *defendant*, *appellant* not getting included in the coreference resolution by the already existing tools.

Additionally, usage of automatic tools to grasp the sentiments in these documents to further analyze the documents in a logical perspective for the purpose of using them as precedents can be considered as an emerging area with significant importance. In the study by Gamage et al.[12] to develop an automatic sentiment annotator, the importance of identifying legal parties in a legal document has been described. However, their study only attempts to extract the phrase level sentiments in legal opinion texts. However, the true value of sentiment analysis on legal opinion texts can be exploited only via the party-based sentiment analysis, where the sentiment of a particular argument, fact, or evidence is extracted concerning each party in a case (Petitioner/Defendant). To this regard, identifying the legal parties in a legal document stands as the first step. Therefore, we believe that our research would be of high value in supporting this kind of research.

4.2 Coreference Resolution(Chamodi)

Identifying mentions of the same entity in different forms and different positions in a text can be defined as coreference resolution in simple terms.

Example 4

- Sentence 4.1: *During the plea process, Lee repeatedly asked his attorney whether he would face deportation.*
- Sentence 4.2: *His attorney assured him that he would not be deported as a result of pleading guilty.*

The two sentences in Example 2 are taken out from the *Lee v. United States*[3] where the two sentences appear consecutively. The words *Lee, his, he* in Sentence 2.1 and the words *His, him, he* 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 is required to identify them as the same entity, rather referred to as tokens in Natural Language Processing(NLP) and coreference resolution delivers to that task.Spacy[13] and Stanford NER system[14] can be named as the two most popular tools for Coreference Resolution.

Table 4.1: CoNLL-2012 Shared Task System Results [15]

System	Open			Closed			Official	Final model	
	English	Chinese	Arabic	English	Chinese	Arabic	Score	Train	Dev
fernandes				**63.37**	58.49	**54.22**	**58.69**	✓	✓
björkelund				61.24	59.97	53.55	58.25	✓	✓
chen		**63.53**		59.69	**62.24**	47.13	56.35	✓	×
stamborg				59.36	56.85	49.43	55.21	✓	✓
uryupina				56.12	53.87	50.41	53.47	✓	✓
zhekova				48.70	44.53	40.57	44.60	✓	✓
li				45.85	46.27	33.53	41.88	✓	✓
yuan		61.02		58.68	60.69		39.79	✓	✓
xu				57.49	59.22		38.90	✓	×
martschat				61.31	53.15		38.15	✓	×
chunyang				59.24	51.83		37.02	-	-
yang				55.29			18.43	✓	×
chang				60.18	45.71		35.30	✓	×
xinxin				48.77	51.76		35.51	✓	✓
shou				58.25			19.42	✓	×
xiong	59.23	44.35	44.37				0.00	✓	✓

Table 4.1 shows the system results for the CoNLL-2012 shared task where contestants designed systems for Modeling Multilingual Unrestricted Co-reference in OntoNotes [15]. The score presented in this table is the unweighted average of MUC, B-CUBED and CEAF scores to determine the winning system. *martschart* mentioned in the Table 4.1 refers to the same system of neuralcoref by spacy. The Table 4.2 shows how the Stanford CoreNLP co-reference resolution systems have performed against the same dataset given in CoNLL-2012. The speed

measurements show the average time for processing a document in the CoNLL 2012 test set using a 2013 Macbook Pro with a 2.4 GHz Intel Core i7 processor. Preprocessing speed measures the time required for POS tagging, syntax parsing, mention detection, etc., while coref speed refers to the time spent by the coreference system[16].

Table 4.2: Stanford CoreNLP Co-reference Resolution System Results[16]

System	Language	Preprocessing Time	Coref Time	Total Time	F1 Score
Deterministic	English	3.87s	0.11s	3.98s	49.5
Statistical	English	0.48s	1.23s	1.71s	56.2
Neural	English	3.22s	4.96s	8.18s	60.0
Deterministic	Chinese	0.39s	0.16s	0.55s	47.5
Neural	Chinese	0.42s	7.02s	7.44s	53.9

4.3 Named Entity Recognition(Chamodi)

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, ORG, LOCATION, etc.
Spacy[13] and Stanford NER system[14] can be named as the two most popular tools for NER.

4.4 Subject Identification(Chamodi)

In the context of English grammar, a subject of a sentence or a clause is an entity that usually explains what the sentence is about or what performs the action(if there is any). Identifying the subject in a sentence indirectly can provide knowledge about who has authority over whom or what. This idea is used in our study in identifying the potential legal parties where we award a certain set of co-ref mentions a score in an instance where it acts as the subject of a sentence or a phrase. Dependency parsing plays a significant role in achieving the goal of extracting the subject from a sentence/phrase. Work on this area has been carried out by Chen and Manning in the research of which is used in building the dependency parser for Stanford CoreNLP [17].

4.5 Recurrent Neural Networks(Melonie)

Recurrent Neural Network (RNN) is a class of artificial neural networks that are best suited for dealing with sequential data. RNN's ability to process inputs of variable length has helped it to show excellent performance in NLP related tasks [18]. Gated Recurrent Unit(GRU) [19] or Long Term Short Term Memory(LSTM) [20] cells can be used to retain the previous hidden state and the current input in RNNs and 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 the Experiments section.

4.6 Sequence to Sequence Learning with Neural Networks(Melonie)

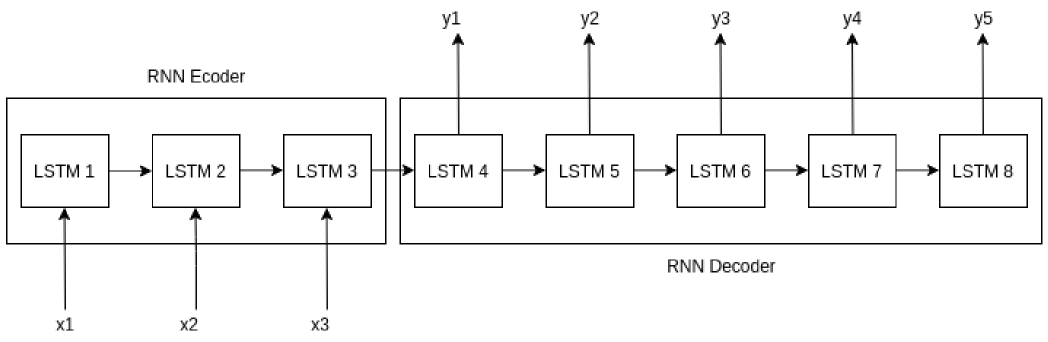

Figure 4.1: RNN Encoder Decoder[21]

Even though deep neural networks have excellent performance in learning difficult tasks when large data sets are available, they cannot be used to generate sequences using input sequences. A multilayered Long Short-Term Memory (LSTM) such as the Recurrent Neural Network(RNN) Encoder Decoder shown in Fig. 4.1, can be used to map the input sequence to a fixed size vector and another multilayer LSTM can be used to decode the output sequence from the vector. LSTM is capable of learning on data with long-range temporal dependencies[21].

4.7 Bidirectional Recurrent Neural Networks(Melonie)

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 et al.[22] 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.

4.8 Word Embedding(Chamodi)

Word embedding is a technique which is used to convert words from a given domain to vectors to create a model that has a distributed representation of words. These techniques are used for natural language modeling and feature learning. Each word of a given text document is mapped to a vector space with its meaning. Also the relationships between words are also mapped to the same vector space. word2vec 2 ,GloVe , and Latent Dirichlet Allocation are the main Word Vector Embedding systems. Both Word2vec [23] and GloVe [24] use word to neighboring word mapping to learn dense embeddings. For this task a neural network based approach is used for Word2vec whereas matrix factorization mechanism is used by GloVe. LDA also has an approach that utilizes matrices but there the words are mapped with the relevant sets of documents. These word embedding methods are useful in our study. Sugatadasa et al in their study Synergistic Union of Word2Vec and Lexicon for Domain Specific Semantic Similarity [25] creates a synergistic union of word2vec, a word embedding method that is used for semantic similarity calculation and lexicon based (lexical) semantic similarity methods and as one of their propositions they show that their method implemented using a legal corpus outperforms the model trained on generic corpus. We take this observation into consideration and wish to try out an implementation of our method basing a word embedding model trained on a legal corpus as well.

In this study, we have incorporated a pre-trained Word2Vec[26, 23, 27] model to create vector embedding of the tokens in the text we use to train our model. We use pre-trained vectors trained on part of Google News dataset where each word is represented by a vector of dimension 300[1].

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

Chapter 5

Methodology

We have taken four approaches in deriving a solution to the problem; namely a rule based approach (Model 1), a sequence to sequence learning approach with character embedding (Model 2) , a BRNN approach with word embedding for word-wise party identification (Model 3) and Utilization of BRNN approach with character embedding for entity-wise party identification (Model 4). From here onward the members of the legal parties involved in the case are referred to as party members.

5.1 Rule based approach for entity-wise party identification(Model 1)

In the Model 1, the legal entities in the legal text are identified using NLP tools along with logical constraints that award each entity with a probability to be an actual legal entity.

The Figure 5.1 shows the overall design of our model that comprises of two different sub models. The first model is *Legal Entity Identification Model*, which does a rudimentary extraction of legal entities from the given legal case. The output from the said model is then passed onto a second model named *Probability Calculation Model*, where the probability for each legal entity to belong to a legal party in the case is calculated.Having calculated the probabilities of all the entities likewise, entities which belong to legal parties are selected out of them based on a pre-decided threshold. Legal party identification model is evaluated with respect to different thresholds using a data set of US supreme court opinions paragraphs with manually identified legal parties.

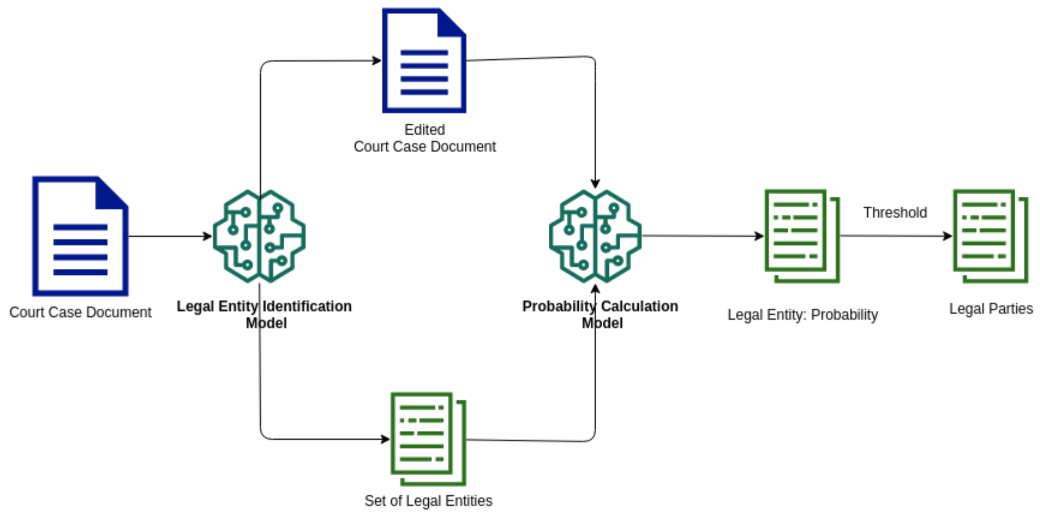
Edited
Court Case Document
Court Case Document
Legal Entity Identification Model
Probability Calculation Model
Threshold
Legal Entity: Probability
Legal Parties
Set of Legal Entities

Figure 5.1: Rule based approach for entity-wise party identification (Model 1)

5.1.1 Legal Entity Identification Model (Chamodi)

This model visualized in Figure 5.2 basically uses co-reference resolution to find the set of entities in the given text and all expressions that refer to the same entity. We use the neural co-reference system in the Stanford CoreNLP to achieve this task due to the proven better performance of the Stanford system in co-reference resolution as shown in the Table 6.11. This set of entities extracted from the co-reference resolution is then run through the next step where the NER test is performed in order to identify the person and organizations entities because only person and organization entities are considered as legal entities. We use the Stanford Named Entity Recognizer [14] since it showcased better performance for the test cases we tested the NER system with as shown in the Table 6.11. The problem we encountered with that system is that it labels each word or rather each token of the given text separately. We came up with a novel method to tackle this issue with a rule based algorithm that is capable of putting together a group of tokens to form a whole of single NER tag. Algorithm 1 describes the above process.

This algorithm basically checks a sequence of tokens(generated by tokenizing a

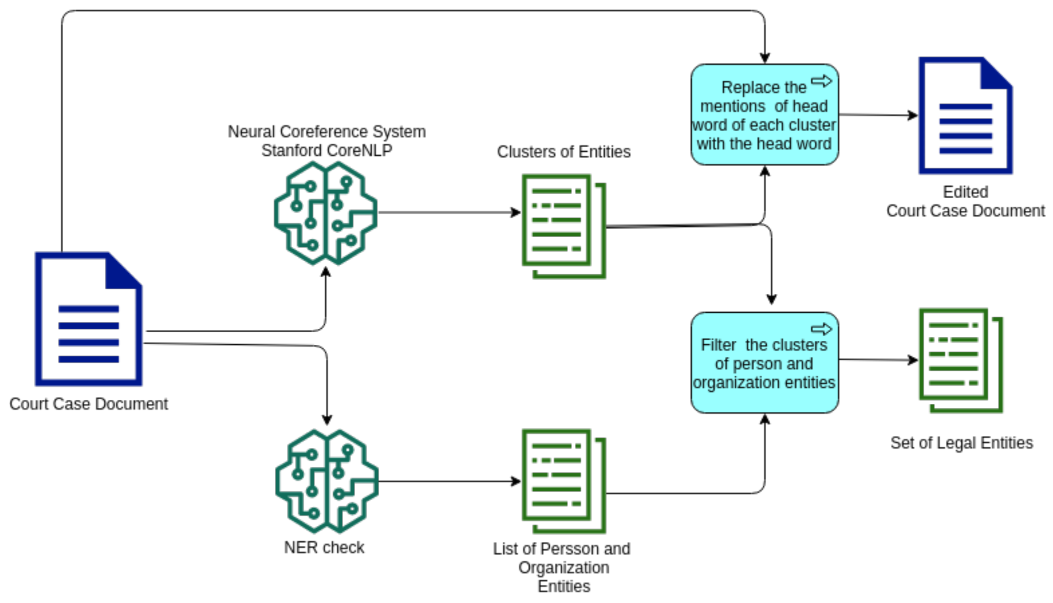
Replace the mentions of head word of each cluster with the head word
Neural Coreference System Stanford CoreNLP
Clusters of Entities
Edited Court Case Document
Filter the clusters of person and organization entities
Court Case Document
Set of Legal Entities
NER check
List of Persson and Organization Entities

Figure 5.2: Legal Entity Identification Model

text using Stanford annotator) for consecutive tokens having the same NER tag, and concatenates them to form a single entity with the same NER. Thus using the output of co-reference resolution model and the NER model, we acquire the co-ref mention sets which are either persons or organizations. The model outputs the head words(that is usually either the mention with the NER tag of PERSON or ORGANIZATION or the first mention of the entity), of these final sets as the set of legal entities.
Additionally, we edit the original court case document with co-reference resolution output for further usage. For that this model replaces all the mentions of a certain co-ref mention set with each of its head word.

5.1.2 Probability Calculation Model(Melonie)

This model visualized as in Figure 5.3 takes the set of PERSON/ORGANIZATION entities and the edited court case document as inputs and outputs the legal entities with the probability of each legal entity to be belong to a legal party of the case. For this task, the model considers the number of times each legal entity is mentioned as a subject of a sentence in the text. The concept behind this method is

Algorithm 1 NER check

Input: token list of a single sentence *tokens*, current word *word*, NER of the current word *NER*

Output: Concatenated tokens of the same NER

```
1: procedure BUILDNER($tokens, word, NER$)
2:     if $tokens[0] == NER$ then
3:         $word = word + \text{BUILDNER}(tokens[1:], tokens[0].word, NER)$
4:     end if
5:     return $word$
6: end procedure
```

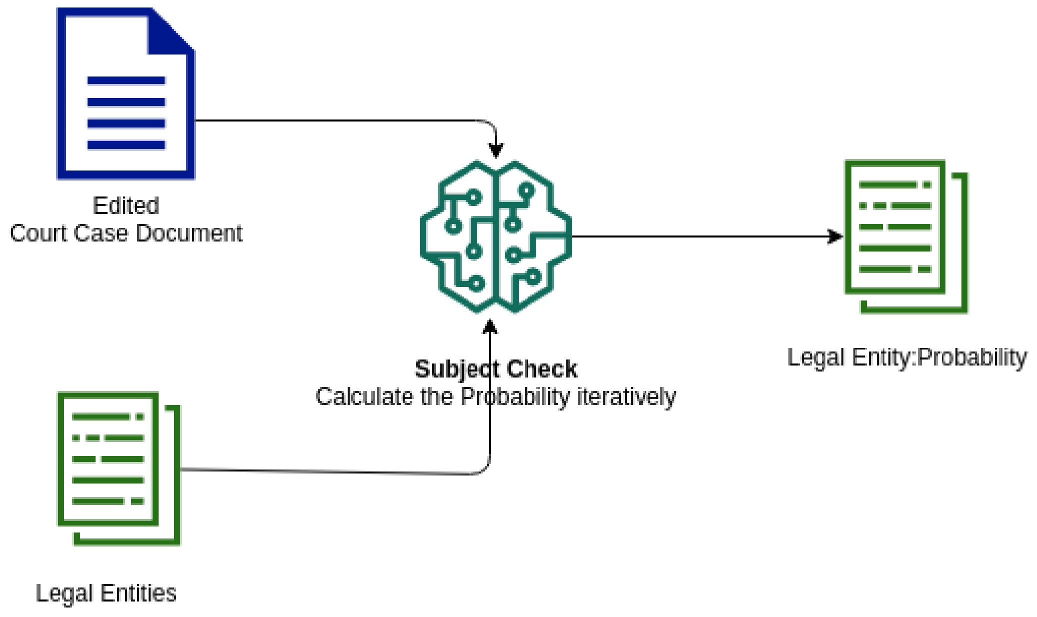

Figure 5.3: Probability Calculation Model

that if a given legal entity is the subject of the most number of sentences, then it is most likely a legal party because it is the main entity that is being discussed about in the document. We can calculate the probability of other legal entities to be in a legal party using the number of times that legal entity has become the subject of a sentence with respect to the maximum number of times a legal entity has become the subject of a sentence.

Output of this algorithm is basically a dictionary in which keys(K=$\{k_1, k_2, ..k_i..k_n\}$)

Algorithm 2 Probability Calculation Model

```
    Input: set of legal entities(K), Edited court case document(D)
    Output: set of legal entities and their probability to be a legal party($k_i : p_i$)
 1: procedure ProbabilityCalcuator($K, D$)
 2:     for each $s_j$ in D do
 3:         $S \leftarrow SubjectOf(s_j)$
 4:         for each $k_i$ in K do
 5:             if $k_i == S$ then
 6:                 $v_i = v_i + 1$
 7:             end if
 8:         end for
 9:     end for
10:     $V \leftarrow Max(v_i)$
11:     for each $k_i$ in K do
12:         $p_i = v_i / V$
13:     end for
        return $k_i : p_i$
14: end procedure
```

are the identified legal entities and the values are the probabilities (P=$\{p_1, p_2, ..p_i..p_n\}$) of those entities to be a legal party. Algorithm 2 explains the process of this model. First this model takes the list of legal entities as keys(k_i) of a dictionary and initializes the current value of each of the key (v_i) to be zero. Then the model takes each sentence (s_j) of the edited document(D) and checks the subject(S) of that sentence (s_j) using Stanford dependency parser [17]. If the subject is a key ($S \in K$) of the input dictionary, it increases the current value (v_i) corresponding to that key (k_i) by one.

$$v_i = v_i + 1 \tag{5.1}$$

Equation 5.1 shows how scores are awarded at a mentioning of an entity as a subject of a sentence where, v_i is the current value of k_i. Likewise every time a key (k_i) appears as the subject of a sentence in the edited document, the current value (v_i) of the key (k_i) increases. After going through all the sentences (s_j) of the edited document(D), this process ends and the probabilities are calculated by dividing current values (v_i) of all the keys(k_i) by the maximum current value (V_i). Then it outputs a dictionary of head words (keys/k_i) and their probability to be an actual legal party (p_i).

$$p_i = v_i/V_i \tag{5.2}$$

Equation 5.2 shows how the final values for the probabilities are calculated where, v_i is the current value of k_i , V_i is the Max(v_i), and p_i is the probability of k_i.

After getting the output from this model where each entity is associated with a probability of being a legal entity, we introduce a threshold value to decide if an entity qualifies to be a legal entity or not. For evaluation purposes we used threshold values of 0.3, 0.4, 0.5, 0.6, 0.7 and 0.8 to see how the system performs with each threshold level.

Example 5

- In fiscal year 2002, petitioner Roberts was injured at an Alaska marine terminal while working for respondent Sea-Land Services, Inc. Sea-Land voluntarily paid Roberts benefits until fiscal year 2005. Roberts then filed an LHWCA claim, and Sea-Land controverted. In fiscal year 2007, an ALJ awarded Roberts benefits at the fiscal year 2002 statutory maximum rate. Roberts sought reconsideration, contending that the award should have been set at the higher statutory maximum rate for fiscal year 2007, when, he argued, he was "newly awarded compensation" by order of the ALJ. The ALJ denied his motion, and the Department of Labor's Benefits Review Board affirmed, concluding that the pertinent maximum rate is determined by the date disability commences.

The final probability output for the text paragraph in Example 4 is as follows: *'petitioner Roberts': 1.0, 'Roberts benefits': 0.15384615384615385, 'respondent Sea-Land Services, Inc.': 0.46153846153846156, 'an ALJ': 0.38461538461538464* Identified legal entities of this text paragraph are Petitioner Roberts, Sea-Land Services and ALJ. With a threshold of 0.3 we can capture all 3 but with a threshold of 0.4, ALJ is no longer identified as a legal entity. So even though we can improve the precision of the system by raising the threshold level, the recall of the system drops as we do so.

5.2 Sequence to sequence learning approach with character embedding for word-wise party identification(Model 2)

In order to overcome the short comings of the above model and to increase the accuracy, we decided to use a deep learning model instead of the rule based Probability calculation model. In the Model 2, we embed the text into a vector using a character embedding method and used sequence to sequence to learning to train a model that is capable of identifying the words that are referred to as legal party.

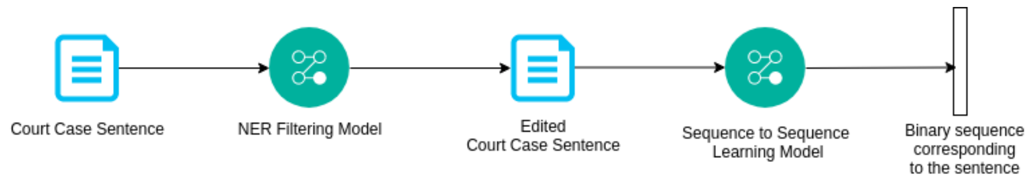
Court Case Sentence
NER Filtering Model
Edited Court Case Sentence
Sequence to Sequence Learning Model
Binary sequence corresponding to the sentence

Figure 5.4: Sequence to sequence learning approach with character embedding for word-wise party identification (Model 2)

The Fig. 5.4 shows the final model which has two sub models, the *NER Filtering model* and the *Sequence to Sequence Learning Model*. *NER Filtering Model* basically takes the sentences in and creates a set of entities that are either a person or an organization. This set is then used to pre-process the sentences where each of the appearances of the identified entities are masked and sent in to the *Sequence to sequence Learning Model*. This is basically an RNN Encoder-Decoder model that would output a sequence of 1s and 0s where each bit implies whether each word in the given word sequence is a legal party or not.

5.2.1 NER Filtering Model (Chamodi)

We use the same method we used to list down the person and organization entities in rule based approach as described in Algorithm 1 for this.Then it edits the original text where each identified entity is masked to be made used of in the RNN model.Each mention is wrapped with a specially generated prefix which contains the information whether it is a person or an organization(*P* for *PERSON* and *O* for *ORGANIZATION*) and is followed by $ to mark the ending of the word. Additionally, the number associated in the prefix helps to identify different mentions of the same entity where the number maps the scattered mentions in to a same cluster. This task is done primarily with the use of co-reference resolution.

Example 6

- Sentence 6.1: *Petitioner Jae Lee moved to the United States from South Korea with his parents when he was 13.*
- Sentence 6.2: *P1\$Petitioner\$ P1\$Jae\$ P1\$Lee\$ moved to the United States from South Korea with P1\$his\$ parents when P1\$he\$ was 13.*

When Sentence 4.1 in the Example 4 is given as the input for this model, it would output Sentence 4.2 where we can observe that all the references to *Petitioner Jae Lee* is masked accordingly.

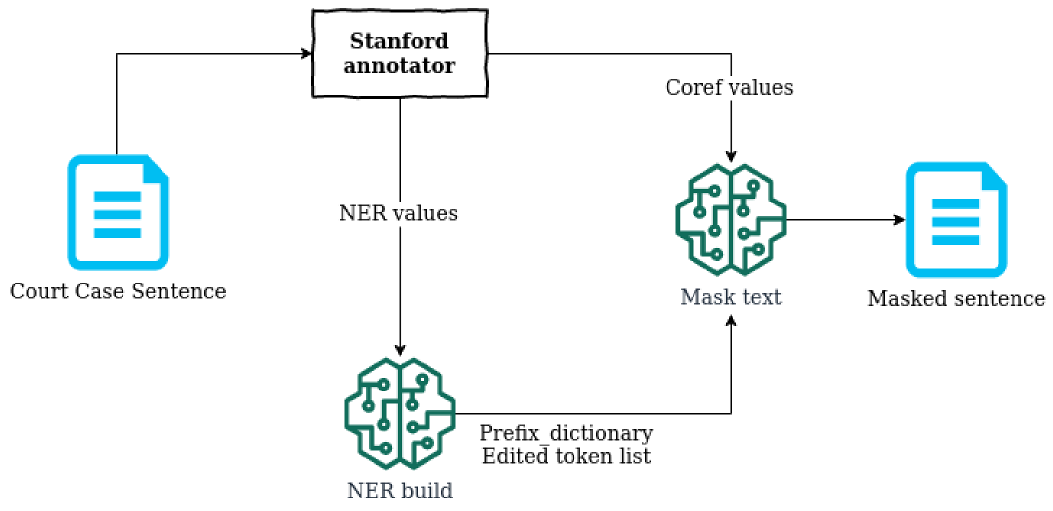

Figure 5.5: NER Filtering Model

Figure 5.5 shows an overview of how the *NER Filtering Model* functions.

5.2.2 Sequence to sequence Learning Model(Melonie)

This model takes the character-wise encoded edited sentences given by the NER filtering model and outputs a binary sequence. Each character (0/1) of the output binary sequence represent whether the each word of the given sentence is a legal party. If the output binary value corresponding to a word of the sentence is 1, it is a legal party. If the output value corresponding to a word of the sentence is 0, it is not a legal party. In this study, we use LSTMs with character embedding method for this task where each character in the text is assigned with a specific index value before sending in for the training. This model consists of two phases as training phase and testing phase.

Training Phase: In this phase,the algorithm takes a set of labeled edited sentences(s_i) given by the NER filtering model. Each sentence has a binary sequence(b_i) where each bit(b_{ij}) implies whether each word(s_{ij}) in the given word sequence(s_i) is a legal party or not. Recurrent Neural Network (RNN) Encoder-Decoder Model(M) is trained with this set of sentences along with binary sequence(b_i) of each sentence(s_i) using stochastic gradient descent(SGD). Then this phase outputs the trained RNN Encoder-Decoder Model($M_{trained}$). We decided to use SGD because it is easier to fit into memory since a single sample is processed at a time and it is computationally fast.

Algorithm 3 Sequence to Sequence Learning Model:Training Phase

```
Input: set of sentences(S) and their output binary sequences(B),RNN Encoder-Decoder Model(M),Number of Epochs(E)
Output: trained RNN Encoder-Decoder Model($M_{trained}$)
1: procedure TrainingM($S,B,M$)
2:     for for $j = 0$, $j{+}{+}$, while $j < E$ do
3:         for each $s_i$ in S do
4:             update parameters of M by SGD for $b_i$
5:         end for
6:     end for
7:     $M_{trained} = M$
8:
9: end procedure
```

- **Testing Phase:** In this phase, the algorithm takes the trained RNN Encoder-Decoder model($M_{trained}$) and a court case sentence(s) as input and it outputs a binary sequence(b), where each bit implies whether each word in the given word sequence(sentence) is a legal party or not.

Algorithm 4 Sequence to sequence Learning Model:Testing Phase

```
    Input: sentence(s), trained RNN Encoder-Decoder Model($M_{trained}$)
    Output: binary sequence of the sentence(b)
1: procedure LEGALPARTYPREDICTOR($M_{trained}, s$)
2:     b=output of $M_{trained}$ for s
3: return b
4: end procedure
```

5.3 BRNN approach with word embedding for word-wise party identification (Model 3)

When we use character embedding to encode the text, the meaning of the word is not specifically taken into the consideration by the system. Therefore we decided to improve the above model with a word embedding method instead of the character embedding method because it might increase the accuracy. In the Model 3, we embed the text into a sequence of vectors using a word2vec and used BRNN to train a model that is capable of identifying the words that are referred to as legal party. Our method consists of four main steps as Tokenizing, Embedding, Masking, and Neural Network Model as shown in Figure 5.6.

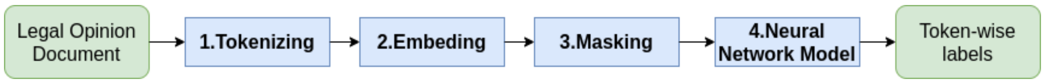

Figure 5.6: BRNN approach with word embedding for word-wise party identification (Model 3)

5.3.1 Tokenizing (Chamodi)

First, the given court case 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, 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 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 masks later.

5.3.2 Embedding (Chamodi)

In this step, a vector for each token that is of dimension 300 is generated with the use of a pre-trained model proposed by Google[1]. In the process of choosing a fitting model for our task, we tested the word embedding model presented along with the work of Sugathadasa et.al [25] on its coverage and also another model trained on Google News dataset, and proceeded with the Google model due to the better coverage it provided even though Sugathadasa et. al model was trained with a domain specific corpus. Table 5.1 presents the coverages of each model for the dataset of 1000 paragraphs we created.

Table 5.1: Coverages of Word Embedding Models

	Google-News	Sugathadasa et. al
available	8474	4569
not available	2909	6814
person/org/location	3387	3387

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 etc.), 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.

5.3.3 Masking (Chamodi)

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 de-

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

ciding this value is explained with the Algorithm 5. 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 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 5.

Example 7

- 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.

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

5.3.4 Neural Network Model (Melonie)

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 must be close to 1 and otherwise it must be close to 0. Figure 5.7 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 text to decide whether any token is referred to a party member or

Algorithm 5 Mask Value Generator

```
    Input: $corefs, NERvaluesofheadwordsofclusters$
    Output: $D_{maskValues}$
 1: procedure ValGenerator($corefs$)
 2:     for each $cluster_k$ in corefs do
 3:         $head_k \longleftarrow cluster_k.headword$
 4:         if $head_k.NER == PERSON$ then
 5:             $val_k \longleftarrow v_1(0 < v_1 < 0.5)$
 6:         end if
 7:         if $head_k.NER == ORGANIZATION$ then
 8:             $val_k \longleftarrow v_2(0.5 < v_2 < 0.75$
 9:         end if
10:         if $head_k.NER == LOCATION$ then
11:             $val_k \longleftarrow v_2(0.75 < v_2 < 1.0$
12:         end if
13:         if $head_k.NER! = PERSONorORGANIZATIONorLOCATION$ then
14:             $val_k \longleftarrow 0$
15:         end if
16:     end for
17:     $D_{maskValues} \longleftarrow setOf(head_k : val_k)$
    return $D_{maskValues}$

18: end procedure
```

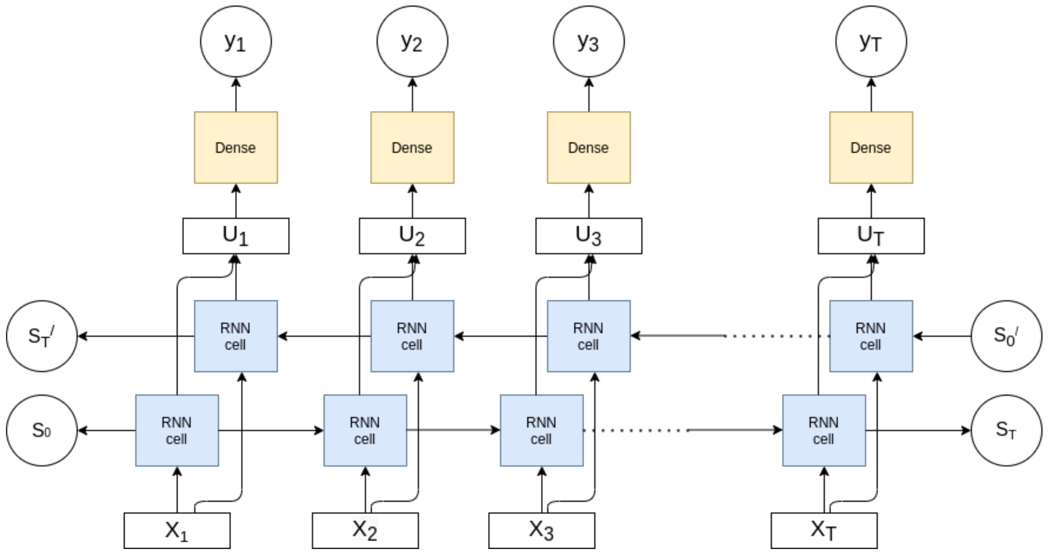

Figure 5.7: Architecture of the Neural Network Model

not. The sequence of output vectors generated by the BRNN layer $\{U_1, ..., U_T\}$ is 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 need to be 1, and outputs corresponding to all the other tokens need to be 0.

Training Phase

In this phase, Neural Network Model is trained to minimize the Binary Cross-Entropy using a dataset 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 basically learns parameters of BRNN and Dense layers to identify tokens of a text sample that are referred 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.

Inference Phase

In this phase, 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 given paragraph. In this phase also, if the number of tokens in a given paragraph is less than T, zero vectors of size n are added to fill the array. 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.

5.4 Utilization of BRNN approach with character embedding for entity-wise party identification (Model 4)

In the Model 4, we modified the Neural Network Model of model 3 (Modified NN model) to identify the words that are referred to as petitioner and defendant separately and trained it to perform word-wise multi-class classification. As shown in Figure 5.8, it consists of 4 main components: Annotator for tokenizing the text and finding the NER and coreference resolution values, Word Vector Generator (W2V) for converting tokens into vectors, Modified NN Model for predicting the probabilities of each word being a mention of either petitioner or defendant party and Party Extractor(E) to identify the entities belong to petitioner party and defendant party using the output of the Modified NN Model.[2]

5.4.1 Annotator (Chamodi)

Annotator is based on Stanford annotator [28] which is widely used. It tokenizes the given text into a sequence of tokens: $\{token_1, ..., token_T\}$ and if the number of tokens present in the given text is less than T (the maximum number tokens

[2]https://drive.google.com/drive/folders/17WCw2X-UuBI51J5yTU3FhuOxhb6TeWf9?usp=sharing

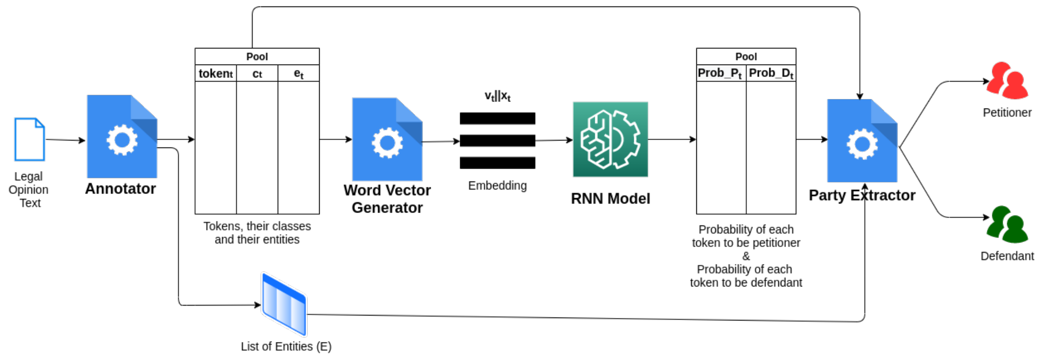

Figure 5.8: Utilization of BRNN approach with character embedding for entity-wise party identification (Model 4)

present in a text sample in the dataset), the text is padded with required number of 0s. Then it identifies the entity (E) of each token: $\{e_1,...,e_T\}$ using coreference resolution and identifies the class (C) of each token: $\{c_1,...,c_T\}$ using NER. Tokens are either words or numbers or punctuation marks. E is the set of identified entities $(entity_1,...,entity_K)$ present in the given text by coreference resolution and C is the set of pre-defined classes (person, organization, locations, None).

5.4.2 Word Vector Generator (W2V) (Chamodi)

W2V is responsible for embedding tokens given by the Annotator and generating word vectors. It utilizes the pre-trained Word2Vec model proposed by Google[3] (Google Word2Vec). Google Word2Vec model generates a vector $\{v_1,...,v_T\}$ of size 300 corresponding to each token. In a scenario where, the model is unable to generate a vector for the token(numbers, punctuation marks, etc.) or the token is referred to a person or an organization, a zero vector of size 300 is returned as the corresponding vector. We avoid generating a vector for person and organization entities because it is not ethical to tune a model that is used for legal tasks for person and organization names. (As an example, in scenarios where the model has been trained for criminal cases, people in other cases who has the same name as the criminals of the training set, would be treated unfairly by the model.)

Afterwards this component adds a value$\{x_1,...,x_T\}$ to the end of each vector $\{v_1,...,v_T\}$ and makes it size 301(n), based on the entity $\{e_1,...,e_T\}$ and the class $\{c_1,...,c_T\}$ of the corresponding token $\{token_1,...,token_T\}$. Tokens belong to the

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

Table 5.2: Value (x_t) that will be added to a vector (v_t) based on its class (c_t)

class (c_t)	c_t=person	c_t=organization	c_t=location	c_t=None
x_t	$0 < x_t <= 0.5$	$0.5 < x_t <= 0.75$	$0.75 < x_t <= 1$	$x_t = 0$

same class (C) are given values in the same range as mentioned in table 5.2 to make sure that the Modified NN Model gives more attention to the tokens belong to person, organization, and location classes because in our implementation, we work under the assumption that only entities belonging to person, organization or location classes can be petitioner or the defendant. This intuitive assumption of ours is also backed by [11]. Also, we give the same value for all the mentions of the same entity(E) according to its class to improve the performance of the Modified NN Model. Algorithm 6 describes this procedure.

Algorithm 6 W2V

```
Input: $E : (entity_1, ..., entity_K), Tokens : \{token_1, ..., token_T\} E_t : \{e_1, ..., e_T\}, C_t : \{c_1, ..., c_T\}$
Output: $Vectors : \{v_1, ..., v_T\}$
1: procedure W2V($E, Tokens, E_t, C_t$)
2:     for each $token_t$ in Tokens do
3:         $v_t \longleftarrow GoogleWord2Vec(token_t)$
4:         $entity_k \longleftarrow e_t$
5:         if $c_t == person$ then
6:             $x_t \longleftarrow (0 + ((0.5 \times k) \div K))$
7:         end if
8:         if $c_t == organization$ then
9:             $x_t \longleftarrow (0.5 + ((0.25 \times k) \div K))$
10:        end if
11:        if $c_t == location$ then
12:            $x_t \longleftarrow (0.75 + ((0.25 \times k) \div K))$
13:        end if
14:        if $c_t == None$ then
15:            $x_t \longleftarrow 0$
16:        end if
17:        $v_t \longleftarrow v_t || x_t$
18:    end for
19: end procedure
```

5.4.3 Modified NN Model (Melonie)

Modified NN Model is responsible for performing token wise multi-label classification, given the sequence of vectors $\{v_1, ..., v_T\}$ generated by the W2V. It basically outputs the probability of each token $\{token_1, ..., token_T\}$ to be a mention of a member of the petitioner $\{prob\_p_1, ..., prob\_p_T\}$ and the probability of each token to be a mention of the defendant $\{prob\_d_1, ..., prob\_d_T\}$ involved in the case. We used multi-label classification instead of multi-class classification because, there can be some neutral entities which appear to be the petitioner and the defendant both in the same case.
Figure 5.9 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.
Modified NN Model has two phases: Training Phase and Test Phase.

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. 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).

Test Phase

In this phase, the algorithm takes the sequence of vectors $\{v_1, ..., v_T\}$ generated by the W2V 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 $\{prob\_d_1, ..., prob\_d_T\}$.

5.4.4 Party Extractor (Melonie)

The purpose of the Party Extractor is to predict the entities (from E) that belong to the petitioner party and defendant party involved in the given legal opinion text using the token wise probabilities: $\{prob\_p_1, ..., prob\_p_T\}$,$\{prob\_d_1, ..., prob\_d_T\}$, given by the trained Modified NN Model for its sequence of vectors generated by W2V and the corresponding entities of tokens E_t:$\{e_1, ..., e_T\}$.
This component takes all the tokens of each entity $(entity_1, ..., entity_K)$ present

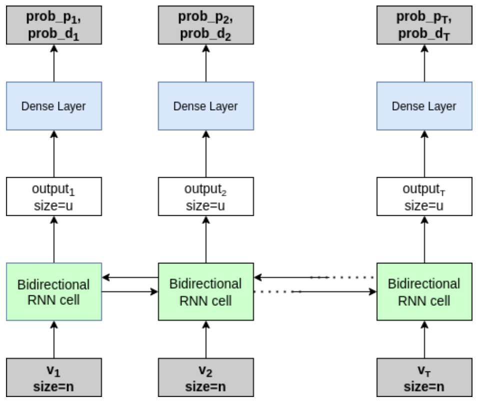

Figure 5.9: Architecture of the Modified NN Model

in the given text and calculates the average of the probabilities of them being a mention of the petitioner party($avg\_p_1, ..., avg\_p_K$) party and the average of the probabilities of them being a mention of the defendant party ($avg\_d_1, ..., avg\_d_K$) separately. if $avg\_p_k >= 0.5 \wedge avg\_d_k < 0.5$, then the $entity_k$ is taken as a petitioner party member and if $avg\_d_k >= 0.5 \wedge avg\_p_k < 0.5$, then the $entity_k$ is taken as a defendant party member. Algorithm 7 describes this procedure.

Algorithm 7 Party Extractor

```
    Input: $E : (entity_1, ..., entity_K), E_t : \{e_1, ..., e_T\}, P_t : \{prob\_p_1, ..., prob\_p_T\}, D_t : \{prob\_d_1, ..., prob\_d_T\}$
    Output: $Petitioner, Defendant$
 1: procedure FINDPARTIES($E, E_t, P_t, D_t$)
 2:     for each $entity_k$ in E do
 3:         $count_k \longleftarrow 0$
 4:         $total\_p_k \longleftarrow 0$
 5:         $total\_d_k \longleftarrow 0$
 6:     end for
 7:     for each $e_t$ in $E_t$ do
 8:         $entity_k \longleftarrow e_t$
 9:         $count_k += 1$
10:         $total\_p_k += prob\_p_t$
11:         $total\_d_k += prob\_d_t$
12:     end for
13:     for each $entity_k$ in E do
14:         $avg\_p_k \longleftarrow (total\_p_k / count_k)$
15:         $avg\_d_k \longleftarrow (total\_d_k / count_k)$
16:         if $avg\_p_k >= 0.5 \wedge avg\_d_k < 0.5$ then
17:             $Petitioner \longleftarrow Petitioner \cup \{entity_k\}$
18:         end if
19:         if $avg\_d_k >= 0.5 \wedge avg\_p_k < 0.5$ then
20:             $Defendant \longleftarrow Defendant \cup \{entity_k\}$
21:         end if
22:     end for
23: end procedure
```

Chapter 6

Experiments

6.1 Rule based approach for entity-wise party identification (Model 1)

6.1.1 Setup

Natural Language Software: All experiments are run using the Stanford CoreNLP tools. Tools for NER, co-reference resolution and dependency parsing were specifically used within our models to come up with the presented results.

6.1.2 Dataset (Chamodi,Melonie)

We handpicked paragraphs consisting of 2-5 sentences and created our own database of 100 such test cases to test against our system. This paragraph set accounted for nearly 200-300 separate legal parties that were used to test the system. The legal cases from which the paragraphs were taken out are from the dataset presented in the paper Synergistic Union of Word2Vec and Lexicon for Domain Specific Semantic Similarity[25]. It contains a large legal data text corpus, several word2vec embedding models of the words in the said corpus, and a set of legal domain gazetteer lists. We used the legal data text corpus in the data set to extract paragraphs and created the test cases, 100 in number which is used to measure the performance of our model.

6.1.3 Performance

We tested this system for different thresholds and the results are as shown in the Table 6.1. A,P,R and F1 are defined with the count of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) of the predicted results as equations 6.1 6.2, 6.3, and 6.4.

$$A = (TP + TN)/(TP + TN + FP + FN) \tag{6.1}$$

$$P = TP/(TP + FP) \tag{6.2}$$

$$R = TP/(TP + FN) \tag{6.3}$$

$$F1score = 2 * P * R/(P + R) \tag{6.4}$$

Table 6.1: System Performance for Different Thresholds

Threshold	P	R	F1
0.3	0.84	0.68	0.75
0.4	0.87	0.63	0.73
0.5	0.88	0.62	0.72
0.6	0.88	0.56	0.69
0.7	0.90	0.54	0.67
0.8	0.89	0.52	0.65

These values are obtained by running text paragraphs through our system. Figure 6.1 shows a comparison of the precision, recall and F1 scores for different threshold levels. It clearly shows how the recall drops when the threshold is increased and how the precision increases when the threshold is increased. The reason as if to why the recall drops is that as the threshold is increased, the system looks for a higher probability value in each legal entity to make the final call of whether it belongs to a legal party or not.F1 score goes to its maximum when the threshold is 0.7.

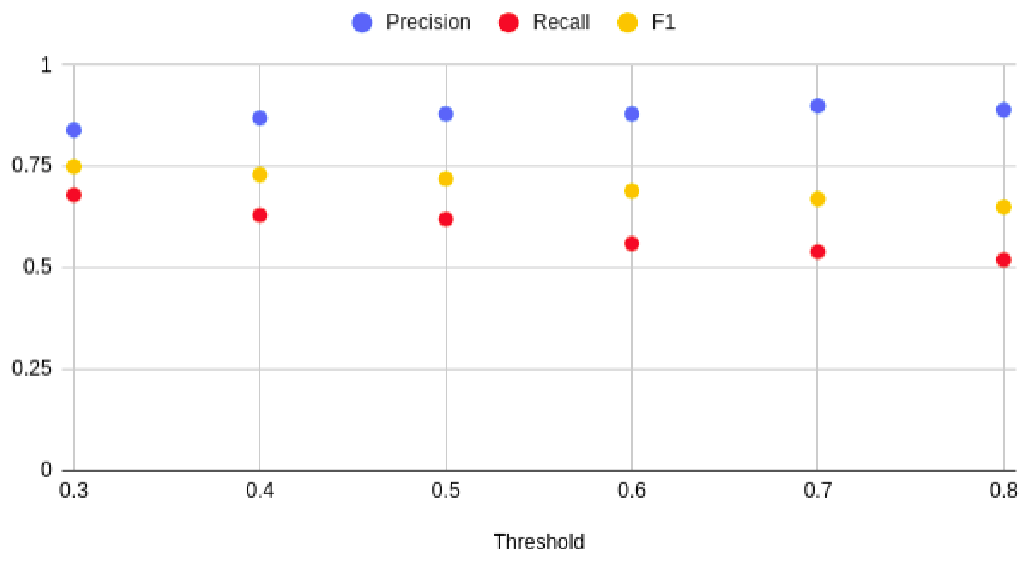

Figure 6.1: Precision, Recall and F1 vs Threshold

6.2 Sequence to sequence learning approach with character embedding for word-wise party identification(Model 2)

6.2.1 Setup

Natural Language Software: All experiments are run using the Stanford CoreNLP tools. Tools for NER, co-reference resolution and dependency parsing were specifically used within our models to come up with the presented results.
Deep Learning Software: Tensorflow python library to implement the Sequence to Sequence Learning Model.

6.2.2 Dataset (Chamodi,Melonie)

We handpicked 200 sample sentences from actual legal documents and manually annotated them with output sequences of 1s and 0s. These annotated sentences were first masked using the NER Filtering Model where the text was edited to mask the words that were either a person or an organization. The words such as he, him, it, they were also masked if they corefed back to person or organization entities. The dataset was sliced as in the Table 6.2 for the training.

Table 6.2: Dataset Slicing

Task	Percentage
Training	0.8
Optimization	0.1
Testing	0.1

6.2.3 Performance of the Sequence to Sequence Learning Model

We trained and tested Sequence to Sequence Model separately for a masked dataset and a non-masked dataset and calculated precision, recall and F1 scores for a test set of 20 sentences and 20 sentences from the training set . Table 6.3 shows the performance of the Sequence to Sequence Model.As seen in Table 6.3, the sequence to sequence learning model trained with the masked dataset showed better performance. We identified the reason for this as the model's ability to quickly learn that person and organization entities have a better probability in being legal entities. Since the model reads the inputs as a character sequence, we believe it identifies the special characters we introduced as masking and use that information for the learning.

Table 6.3: Sequence to sequence Model Performance

	Instances of Training set			Instances of Test set		
	P	R	F1	P	R	F1
non-masked(256 layers)	0.57	0.6	0.58	0.33	0.39	0.38
masked(256 layers)	0.9	0.81	0.85	0.63	0.33	0.43

6.2.4 Performance of the NER Filtering Model

Also we tested the ability of NER Filtering Model to accurately mask the person and organization entities separately for 50 sentences picked from legal documents against the expected masked sentences with precision, recall and F1 scores . Table 6.4 shows the performance of the NER Filtering Model.

Table 6.4: NER Model Performance

	P	R	F1
Person Entities	0.93	0.80	0.86
Organization Entities	0.93	0.91	0.92

6.3 BRNN approach with word embedding for word-wise party identification (Model 3)

6.3.1 Setup

Natural Language Software: Stanford CoreNLP tools were used to perform NER and coreference resolution and for stemming purposes, the Porter stemmer by nltk[29, 30] was used.
Deep Learning Software: We used Keras[31] open source library which runs on top of Tensorflow python library to implement the Neural Network Model
Word Embedding Software: We used a pretrained word2vec model by Google to generate vectors in the Embedding step.

6.3.2 Dataset (Chamodi,Melonie)

We created a list of 1000 sample paragraphs (samples) that are picked from legal opinion texts and tokenized each sample using the tokenizing method we explained in the methodology. Then we manually labeled each token of each sample as a party member mentioned in each sample or not with the help of an expert of the legal domain. Then we generated masked word embedding of each sample by following the embedding and masking steps. The statistics of our dataset is as mentioned in the Table 6.5. The number of tokens that are referred to as party members are just about 3.46% of the total number of tokens of a paragraph.

Table 6.5: Statistics of the Dataset

Attribute	Train	Val	Test	Fullset
Cases	810	90	100	1000
Tokens	351K	39K	43K	433k
Party member tokens	12.16K	1.21K	1.62K	14.99K

6.3.3 Performance of the Neural Network Model

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 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 6.5, the number of expected positives is much less than the number of expected negatives. Performance of the model according to accuracy, precision, recall, f1 score, and the average training time per step is shown in Table 6.6. Training times shown in this table 6.6 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, Precision, Recall, and 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.

Table 6.6: Performance of the Neural Network Model

		Performance				
BRNN cell type	Output units	A (%)	P (%)	R (%)	F1 (%)	Training time per step (s)
GRU	8	97.74	70.38	44.78	54.73	0.42
	32	98.41	78.95	58.15	66.97	0.63
	64	98.95	79.80	73.06	76.28	1.00
	128	99.10	79.49	79.21	79.35	2.00
	256	99.56	86.46	86.56	86.51	4.00
	512	99.88	90.89	91.69	91.29	15.0
LSTM	8	97.58	67.77	34.58	45.79	0.43
	32	98.37	74.77	58.99	65.95	0.73
	64	98.72	81.28	63.90	71.55	2.00
	128	99.34	85.02	79.71	82.28	3.00
	256	99.52	87.50	84.30	85.87	7.00
	512	99.82	89.70	92.16	90.91	20.0

6.4 Utilization of BRNN approach with character embedding for entity-wise party identification (Model 4)

6.4.1 Setup

Natural Language Software: Stanford CoreNLP tools to perform tokenizing, NER and coreference resolution.
Deep Learning Software: Keras [31] open source library which runs on top of Tensorflow python library to implement the Modified NN Model.
Word Embedding Software: Pretrained word2vec model by Google to generate vectors in the W2V.

6.4.2 Dataset (Chamodi,Melonie)

We collected legal opinion documents of 750 different legal cases written in English Language from Synergistic union of word2vec and lexicon for domain specific semantic similarity [25] and picked 750 meaningful sample paragraphs from them[1]. If the number of tokens of a sample paragraph is less than 443 (T, the number of tokens of the paragraph, with the maximum number of tokens), we padded it with the required number of 0s. Then we tokenized each paragraph using Stanford Annotator [28] and an expert of the legal domain manually labeled each token as a mention of the petitioner($prob\_pt = 1$) or not($prob\_pt = 0$) and as a mention of the defendant($prob\_dt = 1$) or not($prob\_dt = 0$). Statistics of the dataset is as shown in Table 6.7.

Table 6.7: 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

[1]https://drive.google.com/drive/folders/1y8J5gDAM7lwEiT8ULWosyGwe4jdfuDM8?usp=sharing

6.4.3 Performance of the Modified NN Model

We trained the 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 6.3) because there is a tendency for the model to generate false negatives as the number of tokens that are mentions of the petitioner/the defendant are only 1.75% / 1.62% of the total number of tokens present in the legal opinion text according to the statistics of the dataset (Table 6.7).
Table 6.8 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 Traning time per step (time) are used as performance measures, taking 0.5 as the threshold.

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 LSTM cells of 32 output units (LSTM-32) has the highest of f1 score for identifying the mentions of the petitioners and RNN Model with GRU cells of 256 output units (GRU-256) has the highest of f1 score for identifying the mentions of the defendants. Therefore we decided to proceed with the experiments of the entire methodology using these two trained RNN Models.

Table 6.8: Performance of the Modified NN Model

BRNN cell type	Output units	Petitioner				Defendant				
		A(%)	P(%)	R(%)	F1(%)	A(%)	P(%)	R(%)	F1(%)	Training time per step(s)
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
	512	99.9	89.0	89.0	89.0	99.9	75.0	75.0	75.0	5
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

6.4.4 Performance of the Methodology for Petitioner and Defendant Extraction

We experimented the performance of our methodology, with selected 2 RNN models (GRU-256 , LSTM-32). Table 6.9 shows the performance for 50 paragraphs of 50 different cases. Precision (P), recall (R) and F1 score are used as performance measures. We can see that the effect of mislabeled tokens is reduced by taking the average because the F1 scores of the overall system are higher than the RNN Model. GRU-256 and LSTM-32 have shown the same f1 score in identifying defendant tokens while GRU-256 has shown a higher f1 score than LSTM -32 in identifying petitioner.

Table 6.9: Performance of the methodology for petitioner and defendant extraction

RNN Model	Petitioner			Defendant		
	P	R	F1	P	R	F1
GRU 256	0.58	0.65	0.61	0.46	0.56	0.51
LSTM 32	0.54	0.65	0.59	0.52	0.49	0.51

Table 6.10 shows the the average run-time for a single paragraph according to the performance of Intel Xeon Processor with two cores @ 2.30 GHz and 13GB RAM.

Table 6.10: Average runtime of each components

Component	Average runtime for a paragraph of 443 tokens(s)
Annotator	4.5
Word Vector Generator	0.7
RNN Model (inference)	2.8
Party Extractor	3.0

6.5 Tool Comparison

6.5.1 Co-reference Resolution Comparison

To get a clear comparison between the two systems on how they perform specifically on our domain, we decided to carry out an initial evaluation on our own. For this we chose a set of sentences taken out from court cases and performed co-reference resolution on them with both of the systems. Table 6.11 shows the results we got from the initial evaluation.

Table 6.11: Stanford vs Spacy Co-reference Resolution and NER Performance

	NER		Co-reference	
	Stanford	Spacy	Stanford	Spacy
P	1.00	0.94	0.93	0.98
R	0.95	0.94	0.92	0.77
F1	0.97	0.94	0.92	0.86

Since the F1 score of the Stanford system came out to be higher than of spacy, we decided to use Stanford system for our co-reference resolution purposes. We also took the recall into consideration since in our system co-reference is used for the preliminary stage of extracting entities. The entities recognized by co-referencing is further filtered out in the next steps, and therefore we believe having a higher recall rather than precision is not harmful on this level. Figure 6.2 visualizes the difference between the performance of the two systems in terms of precision, recall and F1 score.

6.5.2 NER Comparison

We carried out an evaluation similarly for NER on Stanford and spacy systems. The results turned out to be as shown in the Table 6.11.

As visualized in the Figure 6.3, Stanford performed better with respect to all three measurements. And most importantly Stanford had an excellent precision which is crucial for us in our implementation.

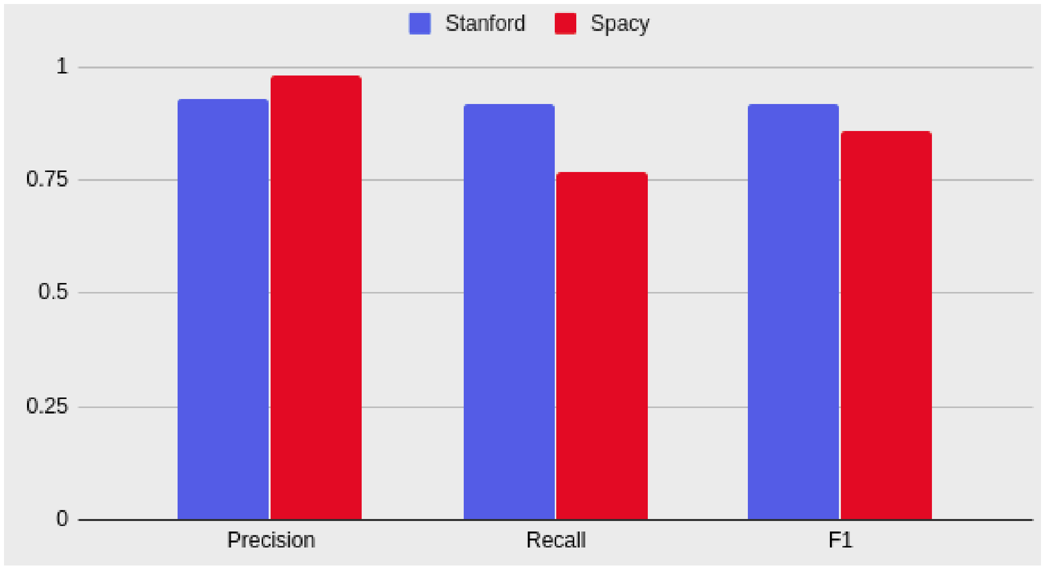

Figure 6.2: Stanford vs Spacy Co-reference Resolution Performance Comparison

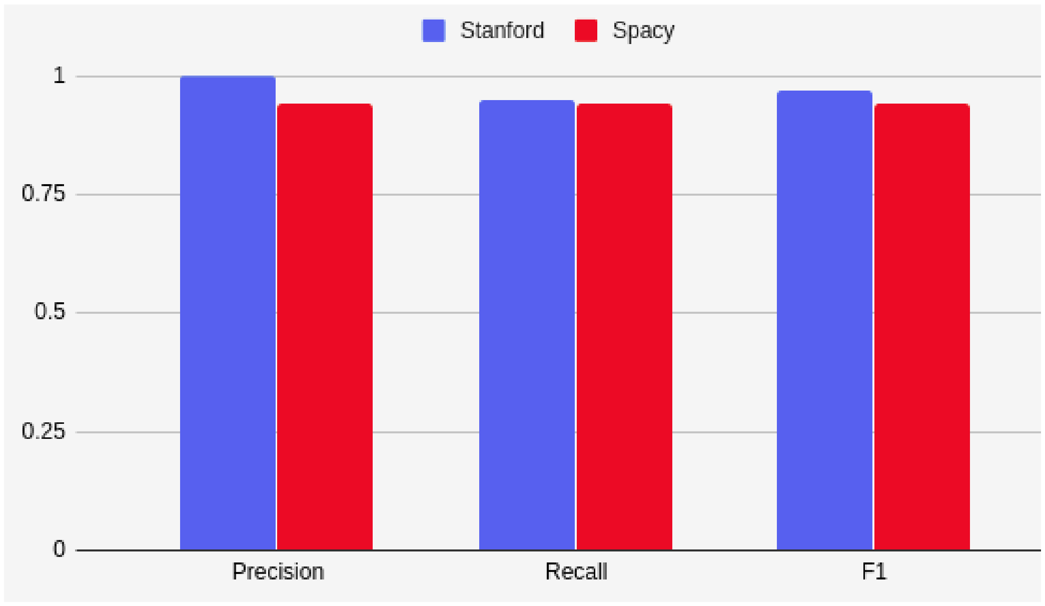

Figure 6.3: Stanford vs Spacy NER Comparison

Chapter 7

Research Paper Submissions

7.1 Party Identification of Legal Documents using Co-reference Resolution and Named Entity Recognition

Conference : International Journal on Advances in ICT for Emerging Regions (ICTer) 2020

Abstract : In the field of natural language processing, domain specific information retrieval using given documents has been a prominent and ongoing research area. The automatic extraction of the legal parties involved in a legal case has a significant impact on the proceedings of legal cases. This is a study proposing a novel way to extract the legal parties involved in a given legal document. The motivation behind this study is that there is the absence of a proper automated system to accurately identify the legal parties in a legal document. We combined several existing natural language processing annotators together with a sequence to sequence learning model to achieve the goal of extracting legal parties in a given court case document. Then, our methodology was evaluated with manually labeled court case sentences. The outcomes of the evaluation demonstrate that our system is successful in identifying legal parties.

State of the paper : Published

7.2 Legal Party Extraction from Legal Opinion Text with Sequence to Sequence Learning

Conference : 15th (IEEE) International Conference on Industrial and Information Systems (ICIIS) 2020
Abstract : In the field of natural language processing, domain-specific information retrieval using given documents has been a prominent and ongoing research area. Automatic extraction of the legal parties (petitioner and defendant sets) involved in a legal case has a significant impact on the proceedings of legal cases. This is a study proposing a novel way to identify the legal parties in a given legal document. The motivation behind this study is that there are no proper existing systems to accurately identify the legal parties in a legal document. We combined several existing natural language processing annotators to achieve the goal of extracting legal parties in a given court case document. Then, our methodology was evaluated with manually labelled court case paragraphs. The outcomes of the evaluation demonstrate that our system is successful in identifying legal parties.
State of the paper : Published

7.3 Identifying Legal party Members from Legal Opinion Text using Natural Language Processing

State of the paper : Under Review

7.4 Petitioner and Defendant Extraction from Legal Opinion Documents

State of the paper : Under Review

Chapter 8

Conclusion

We present four novel methods for legal party extraction from opinion texts. First, We identify the entities that are either a person or a location, or an organization(legal entities) and spot their mentions in the paragraphs. Our first model assumes that the subject of a sentence has the highest possibility to be a legal party and calculate the probability of each legal entity to be a legal Party. Our second and third models evaluate the likelihood of each such mention to be referred to a legal party by inspecting the meaning of the paragraph using a sequence to sequence learning model with character embedding and a BRNN Model with word embedding respectively. In the fourth model, token-wise multi-label classification takes place to classify tokens into petitioner and defendant parties using a BRNN Model. Then the probability of an entity to be a member of the petitioner party or a member of the defendant party is calculated by taking the average of the probabilities of its mentions. In this study, we propose RNN architectures for all these tasks and we trained and tested them for GRU and LSTM of the different number of output units. Also, we introduce datasets that can be used to train these models. We have proven the success of our methods by performing experiments for our whole methods.

Bibliography

[1] K. Tiwari, "Importance of law in society," *Legal Desire*, 2017. [Online]. Available: https://legaldesire.com/article-importance-of-law-in-society/

[2] M. S. Krass, "Learning the rulebook: Challenges facing nlp in legal contexts."

[3] "Lee v. United States," in *US*, vol. 432, no. No. 76-5187. Supreme Court, 1977, p. 23.

[4] V. Jayawardana, D. Lakmal, N. de Silva, A. S. Perera, K. Sugathadasa, and B. Ayesha, "Deriving a representative vector for ontology classes with instance word vector embeddings," in *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*. IEEE, 2017, pp. 79–84.

[5] V. Jayawardana, D. Lakmal, N. 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*, vol. 10, no. 1, p. 1, 2017.

[6] G. Ratnayaka, T. Rupasinghe, N. de Silva, M. Warushavithana, V. Gamage, and A. S. 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)*. IEEE, 2018, pp. 13–20.

[7] G. Ratnayaka, T. Rupasinghe, N. de Silva, M. Warushavithana, V. S. Gamage, M. Perera, and A. S. Perera, "Classifying sentences in court case transcripts using discourse and argumentative properties," *ICTer*, vol. 12, no. 1, 2019.

[8] K. Sugathadasa, B. Ayesha, N. de Silva, A. S. Perera, V. Jayawardana, D. Lakmal, and M. Perera, "Synergistic union of word2vec and lexicon for

domain specific semantic similarity," in *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*. IEEE, 2017, pp. 1–6.

[9] ——, "Legal document retrieval using document vector embeddings and deep learning," in *Science and information conference*. Springer, 2018, pp. 160–175.

[10] I. Chalkidis and I. Androutsopoulos, "A deep learning approach to contract element extraction." in *JURIX*, 2017, pp. 155–164.

[11] A. Gupta, D. Verma, S. Pawar, S. Patil, S. Hingmire, G. K. Palshikar, and P. Bhattacharyya, "Identifying participant mentions and resolving their coreferences in legal court judgements," in *International Conference on Text, Speech, and Dialogue*. Springer, 2018, pp. 153–162.

[12] V. Gamage, M. Warushavithana, N. de Silva, A. S. Perera, G. Ratnayaka, and T. Rupasinghe, "Fast approach to build an automatic sentiment annotator for legal domain using transfer learning," *arXiv preprint arXiv:1810.01912*, 2018.

[13] M. Honnibal and I. Montani, "spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To appear*, vol. 7, no. 1, 2017.

[14] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005, pp. 363–370.

[15] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang, "Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes," in *Joint Conference on EMNLP and CoNLL-Shared Task*. Association for Computational Linguistics, 2012, pp. 1–40.

[16] "Stanford corenlp - corefannotator." [Online]. Available: https://stanfordnlp.github.io/CoreNLP/coref.html

[17] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 740–750.

[18] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[19] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

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

[21] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[22] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[24] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[25] K. Sugathadasa, B. Ayesha, N. de Silva, A. S. Perera, V. Jayawardana, D. Lakmal, and M. Perera, "Synergistic union of word2vec and lexicon for domain specific semantic similarity," in *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*. IEEE, 2017, pp. 1–6.

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[27] T. Mikolov, W.-t. Yih, and G. 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*, 2013, pp. 746–751.

[28] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, “The stanford corenlp natural language processing toolkit,” in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[29] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”, 2009.

[30] E. Loper and S. Bird, “Nltk: the natural language toolkit,” *arXiv preprint cs/0205028*, 2002.

[31] F. Chollet *et al.*, “Keras,” https://github.com/fchollet/keras, 2015.