Unified Legal Party Based Sentiment Analysis Pipeline

Sahan Jayasinghe, Lakith Rambukkanage, Ashan Silva, Nisansa de Silva, Amal Shehan Perera
Department of Computer Science & Engineering
University of Moratuwa
Email : sahanjayasinghe.17@cse.mrt.ac.lk

***Abstract*—The rapid growth of text corpora across various domains has emerged a need and an opportunity to leverage Natural Language Processing to automate and efficiently streamline tedious manual tasks. Legal domain is one such text rich domain which suffers a rapid growth of text corpora and requirement for natural language processing applications. In the pursuit of automating the prediction of the winning party of a court case among other usages, analysing sentiment in a party wise manner is beneficial for legal professionals. The two main sub-tasks in this process is to identify parties in a court case and afterwards analysing the respective sentiment towards each party. In this study we discuss the unification of two such models capable of doing the two task into a single pipeline to perform party based sentiment analysis efficiently.**

***Index Terms*—Natural Language Processing, Aspect Based Sentiment Analysis, Legal party identification , Legal Domain, Sentiment Analysis**

I. Introduction

Various disciplines and domains have their own inherent reliance on manual text dependant task-flows, that are often tedious, repetitive and exhausting. These types of tasks can be potentially automated by the use of Natural Language Processing (NLP). Two major reasons for these problems to exist are: 1) inherent requirements for explicit human intervention, and 2) the lack of practically applicable techniques. Legal domain, can be considered as such a text rich domain, where often the legal professionals find themselves doing a lot of manual work on a daily basis. By its very nature, the legal domain, relies on and produces, a large number of documents, which require expert knowledge and a considerable time and some of these are highly repetitive but at the same time cognitively demanding. These types of problems cannot be addressed by simple procedural programming, rather they require NLP research and applications. Legal Information Extraction (IE) is a prominent research area today since there is a demand to automate the aforementioned tasks for efficiency and the convenience of legal professionals and even the general public.

A. Legal parties

In a legal case, a party consists of a person, a group of people and organizations[1]. In general, court cases have two main parties. In civil cases the party filing the complain is known as the *plaintiff* and the other party is known as the *defendant*. In criminal cases, one party is the *prosecutor* which is a government entity who files the case and the other party is called the *accused* or the *respondent*. There maybe other persons or entities mentioned in a court case, who do not belong to any of these parties, which can be considered third parties, such as witnesses. Also the two parties may be referred to in different non uniform names throughout the legal documents. Thus, identifying the parties and then assessing the impact on the court case with respect to facts and evidence provided by each of the parties, is not a trivial task.

B. Case Law

Case Law can be elaborated as the usage of past court case decisions as grounds to assist the decision of an ongoing court case, rather than using relatively abstract constitutions, statutes and regulations [2]. In a way, it provides the practical examples to the applicability, extents and limitations of applying law on similar kinds of cases. The usage and applicability of Case Law differs based on jurisdictions, similarity or uniqueness of the court cases and other factors. Since they contain the summary of a court case, they can be considered as ideal datasets for applying NLP in legal domain related research and applications.

C. Aspect Based Sentiment Analysis (ABSA)

Sentiment analysis is a prominent technique in NLP where the feeling or opinion of the text content is extracted from textual data [3]. Aspect Based Sentiment Analysis (ABSA) is a specific application of sentiment analysis, where opinion

Correspondence: Sahan Jayasinghe [#1] (E-mail: sahanjayasinghe.17@cse.mrt.ac.lk) Received: 10-08-2022 Revised:30-10-2022 Accepted: 31-10-2022

Sahan Jayasinghe [1] , Lakith Rambukkanage [2] , Ashan Silva [3] , Nisansa de Silva [4] , Amal Shehan Perera [5] are form University of Moratuwa, Department of Computer Science and Engineering. (sahanjayasinghe.17@cse.mrt.ac.lk, rambukkanage.17@cse.mrt.ac.lk, ashansilva.17@cse.mrt.ac.lk, nisansadds@cse.mrt.ac.lk, shehan@cse.mrt.ac.lk).

This paper is an extended version of the paper “Party - based Sentiment Analysis Pipeline for the Legal Domain” presented at the ICTer Conference (2021)

DOI: http://doi.org/10.4038/icter.v15i2.7247

implied by the text content is evaluated and output with respect to specific aspects [4] present in or addressed by the text. In the legal domain this is very useful, since the context in the legal document is often interpreted with respect to the parties of the court case.

D. NLP in the legal domain

Many NLP Researches have been carried out in the legal domain in areas such as: domain specific embedding [5–7], ontologies [7–9], sentiment analysis [10–14], and discourse analysis [15–17] to address aforementioned inefficiencies. Party identification [18–20], which is the recognition of legal parties as introduced in Section I-A, and party based sentiment analysis [12–14], which is the identification of the positive or negative impact of a sentence for party members, are important related researches.

Most of the legal domain related cognitively demanding activities are carried out in a party wise evaluation of natural language statements. For example, Predicting the winning party of a court case is an important use case for legal professionals, as they can use the gained knowledge to drive the court case in their favour by enforcing the important arguments and countering the critical arguments which may be raised by the opposition. Therefore, predicting the winning party of a court case entails the evaluation of arguments in a party-wise manner.

Traditional NLP analysis of the document context as a whole, would not be sufficient since it would be an aggregate representation of the court case rather than an evaluation that takes parties into consideration. For example an argument supporting one party opposes the other party, and this should be captured to in order to evaluate each argument to predict the winning party of a court case. Party based sentiment analysis can be used to approach the problem of predicting the winning party of a court case, since it enables evaluating the context in a party wise approach. In addition, sentiment is a suitable candidate as it is a representation of emotion in the context, which is a valuable feature to be used individually or in combination with other features present in the court case.

Thus far, as stated in the work of Rajapaksha et al. [14], they are the only study to approach the problem of legal party based sentiment analysis. The key bottlenecks we identified in their solution, are two fold:

1) The parties for each sentence have to be given as a manual input
2) Input has to be given as single sentences or small paragraphs.

The first makes expertise in legal domain a requirement, which reduces the automation expected. The second, results in less decisive value, in case of nonuniform representation of parties. The main concern is that for each sentence input, the parties have to be given as manual input for the sentiment to be predicted with respect to each party. For identifying parties in a sentence, expertise in legal domain would be required for nontrivial sentences, and more than that, repetitively inputting parties for each sentence is an exhaustive task and requires a lot of time and manual work. The second concern arising from that is that the input has to be given as single sentences or small paragraphs, with nonuniform representation of parties. This has less decisive value and sentiment of the sentences coherent to a court case with a unique representation of respective parties would be more useful for further analysis.

The focus of our study is to address the above two concerns and reduce manual work. For this, we are driven to use automated party identification. However, it is important to note here that the *Named Entity* of NLP does not directly map to *Legal Entity* (Or *Party*) of the legal domain. since this is conducted in the legal domain, the notion of entities and parties is not consistent with each other (i.e. Only a subset of entities present in a document actually belong to a legal party: petitioner, defendant).

Ex.1 taken from United States Supreme Court, elaborates how basic *Stanford Core NLP* [21] Named Entity Recognition (NER) model tags the entities. But, the location entities *Cupey* and *Puerto Rico* are not *legal parties*. Thus, it was understood that using general purpose NER to identify only the legal entities is not trivial. Therefore, we decided to use the model proposed by de Almeida et al. [20] which is specifically used to identify parties in legal court case documents. By using the output of de Almeida et al. [20] as one of the inputs to Rajapaksha et al. [14], we propose a fully automated pipeline.

Ex.1 : Cao v. Commonwealth of Puerto Rico [22]
Dolores H. Cao, an elderly resident of Cupey, Puerto Rico, was removed from her home, made to undergo a psychological evaluation, and placed in a substitute home and, later, a state institution for the elderly, by the Puerto Rico Family Department (the Department)

Upcoming sections of this paper are organized as follows. Details of previous research works related to our study is discussed in Section II. Methodology and implementation details of the pipeline are elaborated in Section III and the experimental details are listed in Section IV. This journal paper is an extension of our previously published conference paper [23]. We have provided a more detailed discussion of the research work, with different approaches we tried out to come up with the final PBSA pipeline.

II. Related Work

A. Party Identification in the Legal Domain

The study by Samarawickrama et al. [18] is fundamentally based on the frequency at which a given entity occurs as the subject of a sentence. In order to obtain this statistic, they have used two sub-models: 1) use Name Entity Recognition and co-reference resolution to identify entities, and 2) calculate a probability output based on subject frequency. A threshold is then used to identify the entities belonging to a party.

In contrast, the study by de Almeida et al. [19] approaches the problem in a different perspective. First an NER Model

is used for identifying people and organizations, whereupon which a mask is applied on them. Next, co-reference resolution is used to resolve different representations of the same entity. A custom algorithm is used to replace the multi-token representations of entities. Finally, the masked sentences of the NER model is fed to a sequence-to-sequence model. The output binary sequence then discerns whether each word belongs to a party or not.

In their follow up study, de Almeida et al. [20] have introduced a novel method to identify parties more accurately using deep learning. They have trained and evaluated a number of deep learning models experimenting with different architectures of Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) cells for a data set (1000 paragraphs) created using court case documents of US Supreme Court published by Sugathadasa et al. [5]. The process has four steps: 1) Tokenizing, 2) Embedding, 3) Masking, and 4) the Neural Network. NER and co-reference resolution are used to identify entities in the *Tokenizing* step. For *Embedding* they have used a Word2Vec [24] model of 300 dimensions trained on Google News data[1]. The *Masking* step uses the identified entities and brings the dimensions up to 301. Finally, the vector is passed to a *Neural Network* based on Bidirectional Recurrent Neural Networks (Bi-RNN) which outputs the probability of each token belonging to a Legal party.

B. Party Based Sentiment Analysis

In the study by Rajapaksha et al. [12], the sentiment for each party is calculated using sentence level sentiment. To allocate the sentiment to each party they have used a simple convention. They have used the co-reference resolution of Stanford Core NLP [21], to resolve for pronouns and other references of the same entity. Further, they have used the constituency parser of the same to breakdown the phrases for sentiment annotation.

Mudalige et al. [13] describes a solution for a major drawback in NLP sector related to the legal domain which is the unavailability of a public dataset for Aspect (Party) Based Sentiment Analysis. As discussed in Section I-A, a legal case consists of two or more parties, where each party belongs to one of the two categories: *petitioner* or *defendant*. The researchers have worked on developing a dataset with sentiment annotation for each party mentioned in a sentence which supports multiple sub tasks related to ABSA. Further, in a followup study, Rajapaksha et al. [14] have implemented a deep learning model for detecting the sentiments towards each legal entity referenced in a given sentence using the dataset they created [13]. The results achieved have been compared with a number of ABSA models [25–32], all of which has been outperformed by an accuracy of 0.7 and 0.62 F1 score.

III. Methodology

The objective of this study is to define a pipeline which uses the output of party extraction system of de Almeida et al. [20] to generate the input for sentence-wise party based sentiment analysis system of Rajapaksha et al. [14]. Party identification model by de Almeida et al. [20] was selected as they have shown above 97% accuracy for all their model variations and 90.89% precision and 91.69% recall for their best performing model, for the test data set. The PBSA model by Rajapaksha et al. [14] was selected as it has outperformed a number of ABSA models at the time of reference [25–32]. On top of their performance factor, these two subsystems were specifically selected to be combined, as both of these models have been trained on United States Supreme court cases.

The purpose of implementing this pipeline is to eliminate need of manual input of legal entities referenced in the sentence to Party-Based Sentiment Analysis model. To achieve this we have come up with a more accurate solution advanced over inferences gained, with respect to the baseline approach of using the two systems in [14, 20] as is. The evaluation results through the evolution of our pipeline is presented in section IV.

A. Party Extraction System

de Almeida et al. [20] have used a pre-trained Word2Vec model by Google for 300-dimensional word embeddings and implemented a procedure to add a mask value (as the 301st dimension) for *Person*, *Organization* and *Location* entities recognized by Stanford's Named Entity Recognition model.

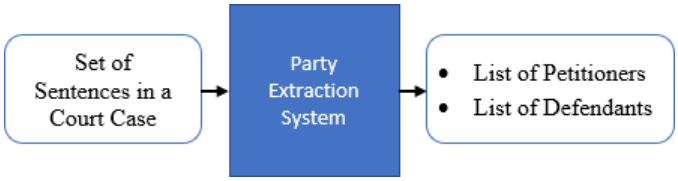

Fig. 1. Party Extraction System

Their best performing model turned out to be the GRU model with 512 output units (referred as *GRU model* in this paper here onward). Using the output sequence of the model, they have implemented the legal entity extraction method with the use of co-reference annotation. As shown in Fig 1, this model serves as the *Party Extraction System* component of our pipeline.

B. Party-based Sentiment Analysis (PBSA) System

The study by Mudalige et al. [13] is based on integrating the ABSA concept into the legal domain. They point out the unavailability of a publicly available dataset for ABSA in legal domain as a main issue. It should be noted that according to Caswell et al. [33], NLP datasets that are publicly available but not tagged by experts may be of insufficient quality. With the help of law experts, Mudalige et al. [13] have created an annotated public dataset for Party-Based Sentiment Analysis which consists of sentences taken from the US Supreme Court document data set released by Sugathadasa et al. [5].

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

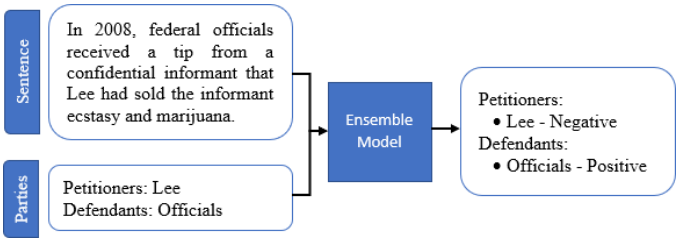

Fig. 2. PBSA System working on a sentence from Lee v. United States [34]

Further, they have included sub-sentences obtained from the above sentences using the Stanford Dependency Parser. These information are given to the system along with the relevant entities and their references for training the deep learning model. For each entity, they have used three labels to represent the sentiment within the context:

- negative : −1
- neutral : 0
- positive : +1

They have concluded after their experiments that the ensemble model implemented using BERT [35] embeddings, Attention-based LSTM with Aspect Embedding architecture (ATAE-LSTM) [26], and the Graph Convolution Networks (GCN) proved to be the best performing model. In this study, we have used this ensemble model to gain the sentiment outcomes for each sentence of a given text with respect to identified legal entities from the party extraction system. As shown in Fig 2, this model serves as the *Party-based Sentiment Analysis (PBSA) System* component of the pipeline described in this work.

C. Pipeline Overview

Fig. 3. Pipeline Overview

In order to output party-based sentiment for each sentence in a given text, we have defined an abstract view of the pipeline proposed by this study in Fig. 3. The basic functionality of the pipeline is to take a court case document as input, and output the party-based sentiment for all the sentences in the document. First we use the Party Extraction system to identify the party members and pass it on through an adapter as an input to the Party-Based Sentiment Analysis system in the expected format. The output of the Party Extraction system after the intermediate adapter, is the petitioner and defendant entities and their references in each sentence of the given text ordered exactly as they appear. This is subsequently fed to the PBSA system, which outputs the party based sentiment for all the sentences.

D. Baseline Model

As the starting point of implementing the pipeline, we straightforwardly used the Party Extraction system defined by de Almeida et al. [20] which consists of a deep learning model which predicts the petitioner and defendant probabilities of each token and a party extraction algorithm which uses Stanford Co-Reference to provide the names of petitioners and defendants. We used the aforementioned existing party extraction algorithm to obtain the party members and input them to the latter half of the pipeline. However, we observed that, for certain scenarios, the extracted names of organizations or persons involved in a specific party deviated from expected string. This deviation came in the form of a description tailing behind the expected string. An example of this faulty co-reference annotation is shown in Fig. 4.

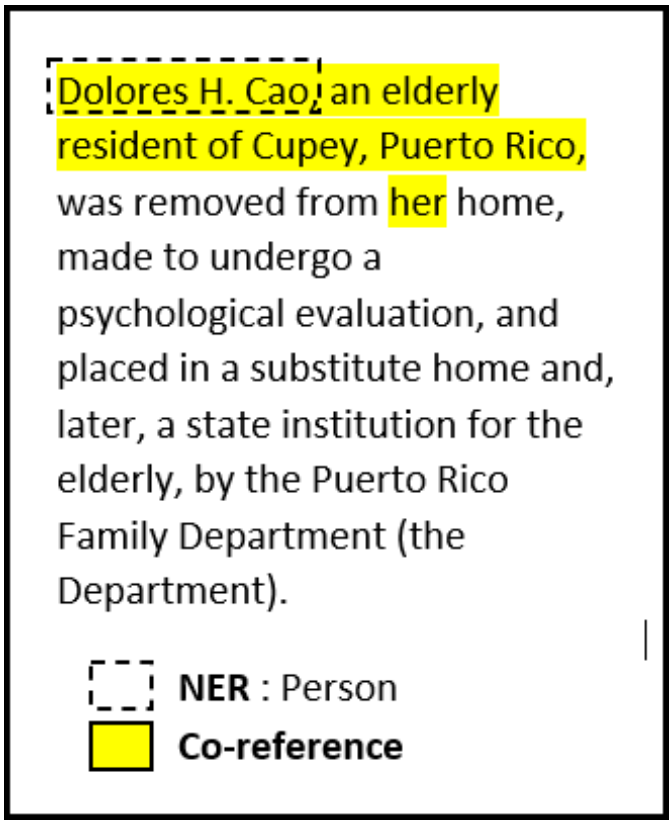

Fig. 4. Example of faulty co-reference annotation in Special Scenarios

In Fig. 4, which uses a sentence obtained from *Cao v. Commonwealth of Puerto Rico* [22], the pronoun *her* should be identified as the co-reference of *Dolores H. Cao*. But instead, the extraction function of Samarawickrama et al. [18] forms

an erroneous entity where the tailing description of *Dolores H. Cao* is also included in the entity.

E. nuRef model: Improving Stanford Co-Reference Output

To mitigate the issue of extracting the entities with a tailing description, we implemented an algorithm to update the co-referenced entities to actual entity name described in Algorithm 1.

Algorithm 1: Co-Reference Update Algorithm

```
Input: List of tokens and Co-reference annotation
       generated by Stanford Annotator
Output: Updated Co-Reference annotations
Function UpdateCoRef(TokenList, CoRef List):
    for each CoRef in CoRefList do
        Entity := CoRef.Entity
        EntityTokens :=
          TokenList[Entity.StartTokenIndex to
          Entity.EndTokenIndex]
        EntityTokensLength := Lenth(EntityTokens)
        for CurrentTokenIndex:= 0 to
          EntityTokensLength do
            CurrentToken :=
              EntityTokens[CurrentTokenIndex]
            if CurrentToken.NER = "PERSON" or
              "ORGANIZATION" or "LOCATION"
              then
                SameNERWords := FindSameNER-
                  Words(CurrentToken.NER,
                  EntityTokens[CurrentTokenIndex :
                  Entity.EndTokenIndex])
                CoRef.Entity.text := SameNERWords
    return CoRef List
end Function

Input: NER value of starting token, List of tokens
Output: Combined words with same NER
Function FindSameNERWords(NER,
  TokenList):
    SameNERWords := List()
    for each Token in TokenList do
        if Token.NER = NER then
            SameNERWords.add(Token.word)
        else
            break
    return String joining each element of
      SameNERWords by space
end Function
```

The function *UpdateCoRef* defined in Algorithm 1 takes the list of tokens generated by the Stanford Annotator for the given text and the co-reference annotation as input. Since the goal of this function is to remove descriptive fields from an entity and replace the respective entity field of *CoRef* object with the actual name of the entity, the execution goes through each co-reference entity in the annotation and incorporates Stanford NER annotation to extract out the actual words representing the entity.

Respective NER values for each token exist as a field named *NER* in the tokens list and to combine the consecutive words with same NER value, we have defined a separate function *FindSameNERWords* in Algorithm 1. We need to pass the NER value which we look for in consecutive words and the token list starting from the current position of the token in *CoRef* entity (which is a sub-list of tokens). The function *FindSameNERWords* is called whenever a token is deemed to be belonging to one of the three categories: *Person*, *Organization*, *Location*. This function finds the consecutive words with same NER value as the current token.

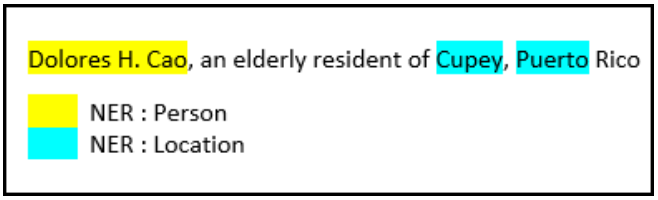

Fig. 5. Co-Reference update using NER (Cao v. CoPR [22])

Fig. 5 shows the result of sending the faulty entity tagged in Fig. 4 through Algorithm 1.Since it finds a token which belongs to *Person* category first, then it searches for consecutive words with the *Person* NER value. As soon as the algorithm encounters a token with a different NER value, it returns the words found so far with the first NER value (in this case, *Person*) immediately and forgoes further scanning through the remaining tokens of the Co-Reference entity. Then the relevant field of the *CoRef* object is updated by combining the same NER words which were returned. In the example given in Fig. 5, it is *Dolores H. Cao*.

We define the *nuRef* model with the GRU of de Almeida et al. [20] which was trained using masked vectors generated using Stanford original Co-Reference annotation and the updated updated co-reference. When the *nuRef* was tested, it was observed that even though the entity strings which are output have improved in quality, the disparity between the pre-trained GRU and the new co-reference has degraded the accuracy which is based on alignment compared to the *Baseline* model discussed in Section III-D. Therefore, it was decided that the GRU model should be re-trained using the masked vectors generated using the updated Co-Reference approach.

F. nuRefGRU model: Training the GRU using the Updated Co-Reference

We followed the same approach proposed by Samarawickrama et al. [18] for generating mask values for entities and their references. However, while the original model uses Stanford NER fed to Stanford Co-Reference, we fed Stanford NER to the updated Co-Reference. Pre-trained Google Word2Vec

model was used to vectorize the tokens and the mask value was added to vectors making them 301-dimensional. We used the same data set released by de Almeida et al. [20] which consists of 1000 court case paragraphs where each token is labeled as *petitioner* or *defendant*.

We define this model with the GRU retrained with updated co-refernce as *nuRefGRU* model. It was observed that the *nuRefGRU* model out-performers not only the *nuRef* model but also the *Baseline* model. Therefore, with the *nuRef-GRU* model we proceeded towards the implementation of the adapter for generating the input for Party-Based Sentiment Analysis system [14].

G. Pipeline Implementation 1

The inputs for the *adapter* set between the *party extraction system* and *PBSA system* are: list of tokens, updated co-references, and the outputs of the *party extraction system*. The latter consists of the list of petitioner entities and the list of defendant entities. The output of the *adapter* is the sentence-wise references of petitioners and defendants which can be used directly as input for PBSA System defined by Rajapaksha et al. [14].

In order to provide the petitioner and defendant entities in the same order they are mentioned in each sentence for Party-based Sentiment Analysis System implemented by Rajapaksha et al. [14], Algorithm 2 populates an object which stores sentence-wise petitioner and defendant entities (also their references like pronouns and surnames for Person entities, abbreviations for Organization entities) as key, value pairs. Key is the token index in the respective sentence and value is the entity itself or the reference words.

At the last section of Algorithm 2, since the party members which are referenced only once in the text are not in Co-Reference annotation, they are searched through the token list and added to the object *LegalEntityPositions*. We can extract the sentences as a list using Stanford Tokenization as well.

At this point, all the data needed as input for the Party-based Sentiment Analysis system are ready. Subsequently we have to iterate through sentences of the text passing the text and respective mentions of petitioner and defendant entities for PBSA system. Workflow of the combined systems with the intermediate adapter is elaborated by Fig. 6.

H. Drawbacks of Implementation 1

Implementation 1 (Section III-G) depends on both Stanford Annotation accuracy and the deep learning model for party probability prediction accuracy. The final algorithm of Party Extraction System defined by Samarawickrama et al. [18] for extracting the entities belonging to petitioner and defendant parties depends on the probabilities (two values for petitioner and defendant) predicted by deep learning model for each token and the detection of Person, Organization and Location entities and their references by the Stanford Annotator.

We analyzed that even though the GRU model provides high probability for an entity in the text, sometimes Stanford's Annotator failed to recognize this as a Person, Organization or Location entity. Due to this conundrum, the party extraction system fails to return the entities recognized only by the GRU model. Also, there were court cases where the petitioner or defendant party is referred more generally without specifying names of the people or organizations belonging to that party (*Ex.2*).

Algorithm 2: Adapter implemented using extracted party members

Input: List of tokens, Updated Co-Reference, List of Petitioners, List of Defendants
Output: Object containing sentence-wise references of petitioners and defendants indexed by token location

```
Function AdapterCoref(TokenList, CoRef List, PetitionersList, DefendantsList):
    LegalEntityPositions := Object()
    for each CoRef in CorefList do
        if CoRef.Entity in PetitionersList then
            for each Reference of CoRef.Entity do
                sentenceIndex := Reference.sentInd
                LegalEntityPositions.sentenceIndex.Petitioner
                  := {TokenIndex: CoRef.text }
            remove Coref.Entity from PetitionersList
        else if Coref.Entity in DefendantsList then
            for each Reference of CoRef.Entity do
                sentenceIndex := Reference.sentInd
                LegalEntityPositions.sentenceIndex.Defendant
                  := {TokenIndex: CoRef.text }
            remove Coref.Entity from DefendantsList
        else
            continue
    if PetitionersList is not empty then
        for each Petitoner in PetitionersList do
            find location of Petitioner in TokensList and
            update LegalEntityPositions
    if DefendantsList is not empty then
        for each Defendant in DefendantsList do
            find location of Defendant in TokensList and
            update LegalEntityPositions
    return LegalEntityPositions
end Function
```

Ex.2 - Tobar v. US [36]
Plaintiffs are Ecuadorian crew members of a fishing boat. The United States Coast Guard saw their boat in international waters near the Galapagos Islands and suspected it of involvement with smuggling drugs. The Coast Guard stopped Plaintiffs'boat and boarded it. Tests performed on the vessel yielded suspicious but inconclusive results and, with the consent of the Ecuadorian government, the Coast Guard towed the boat to Ecuador. Further tests conducted by the Ecuadorian government uncovered no contraband, and no charges were filed against Plaintiffs. Plaintiffsthen sued the United States for damages resulting from these events.

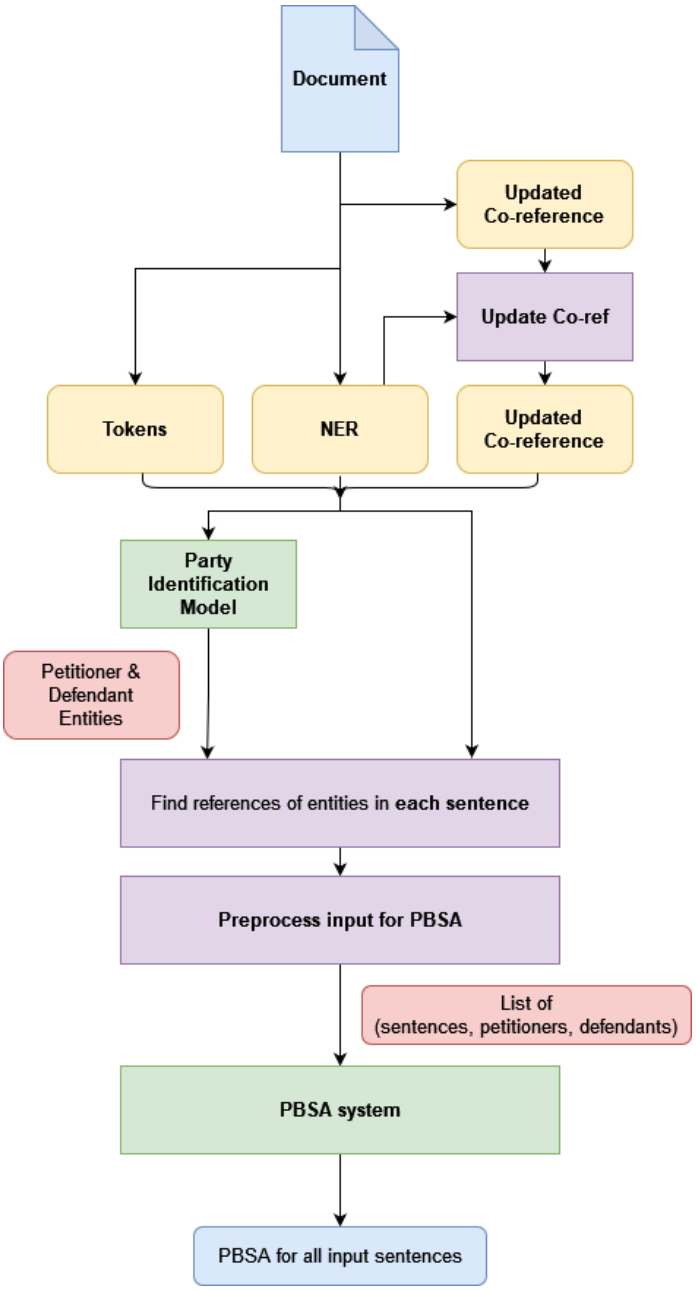

Fig. 6. Pipeline with updated Co-Reference Annotation

When considering similar cases as *Ex.2*, the GRU model has been able to predict high confidence for the word "Plaintiff" as an entity of the petitioner party. But the Stanford Annotator does not recognize those as entities. On the other hand, in situations where the GRU model has not identified a reference of an entity as a petitioner or defendant, the Stanford Annotator could recover those references and include in the output of the adapter connecting to PBSA system.

Since there's both positive and negative aspects of the Implementation 1 (Section III-G), we researched for a different approach to overcome the existing issues.

I. Pipeline Implementation 2

In this implementation, the inputs for the adapter are the petitioner and defendant probability arrays predicted by GRU model for each token of given text input. Using these probability arrays and token list of the court case text, we defined the adapter to generate the input for Party-based Sentiment Analysis system defined by Rajapaksha et al. [14].

Algorithm 3: Adapter implemented using Petitioner & Defendant probability sequence

Input: List of tokens, Sentence start indices of the token list, Petitioner probability list (PPL), Defendant probability list (DPL), probability threshold (Thresh)

Output: List containing sentence-wise references of petitioners and defendants

```
Function AdapterProb(TokenList, SentenceStartIndices, PPL, DPL, Thresh):
    EntityList := List()
    TokenIndex := 0
    for i := 0 : Length(SentenceStartIndices)-1 do
        endInd := SentenceStartIndices[i + 1] if i + 1 is an index in SentenceStartIndices; else Length(TokenList)
        PetitionerTokenWords, DefendantTokenWords := List()
        while TokenIndex < endInd do
            ConditionP := PPL[TokenIndex] ≥ Thresh and DPL[TokenIndex] < Thresh
            if ConditionP then
                PetitionerEntity := Empty String
                while ConditionP do
                    concatenate token word to PetitionerEntity
                    TokenIndex := TokenIndex + 1
                PetitionerTokenWords.add(PetitionerEntity)
            ConditionD := DPL[TokenIndex] ≥ Thresh and PPL[TokenIndex] < Thresh
            else if ConditionD then
                DefendantEntity := Empty String
                while ConditionD do
                    concatenate token word to DefendantEntity
                    TokenIndex := TokenIndex + 1
                DefendantTokenWords.add(DefendantEntity)
            else
                TokenIndex := TokenIndex + 1
        EntityList.add(List(PetitionerTokenWords, DefendantTokenWords))
    return EntityList
end Function
```

Algorithm 3 defines the process to create a 3-dimensional list of party members and their references where the 1st dimension represents the sentence index, 2nd represents whether petitioner or defendant and the 3rd stores the names of the entities or their references in the same order they appear in the respective sentence. This algorithm takes the list of tokens as a 1-dimensional array, and to identify the token that starts

a new sentence, we need to provide the indices of sentence starting tokens in the tokens list. This array can be generated from Stanford Annotation.

We have incorporated a threshold parameter to classify the tokens as either petitioner party or defendant party. When the token has the petitioner probability value greater than or equal to the threshold, and if the defendant probability is less than the threshold, the algorithm extracts that token as a petitioner. The inverse of the said condition extracts the token as a defendant. Other combinations have been neglected for party extraction.

Since the deep learning model is trained for probability prediction by labeling the petitioner and defendant entities and their references as well, probability arrays that we provide for *Algorithm 3* has higher petitioner or defendant probability for tokens representing entity references also. Accuracy metrics for GRU model is presented in the Experiments and Results section. The workflow for the combined systems along with the intermediate adapter we defined in this approach can be elaborated as Fig. 7.

With this implementation of the pipeline, the main issue we faced in the Implementation 1 (Section III-G) is mitigated, since the identification of petitioner and defendant entities along with their references solely depends on the Deep Learning model's output.

There is no impact from the Stanford NER and Co-Reference accuracy for the pipeline in this approach. But, when a reference of a party member is not predicted with a higher probability by the Deep Learning model, there is no way to recover those words to include in the input of PBSA system. That is because there is no co-reference identification unlike in the Implementation 1 (Section III-G).

IV. Experiments and Results

In order to ensure accurate inputs are fed into the Party-Based Sentiment Analysis system implemented by the researchers Rajapaksha et al. [14], we need to evaluate the output of Deep Learning model which predicts the petitioner and defendant probabilities for each token in text. Initially, we used the Bi-directional Recurrent Neural Network model which consists of Gated Recurrent Units with 512 output units, trained and evaluated as the best performing model by researchers Samarawickrama et al. [18] for the dataset of 1000 US supreme court case paragraphs. We compared the accuracy, precision, recall and f1 metrics for the last 100 paragraphs of the dataset. Rows of *Table I* and *Table II* represents the metrics for the following model configurations respectively.

1) **Baseline Model:** GRU 512 model trained and evaluated using original Stanford Co-reference as defined in Section III-D.
2) **nuRef Model:** GRU 512 model trained using original Stanford Co-Reference and evaluated using updated Stanford Co-reference as defined in Section III-E.

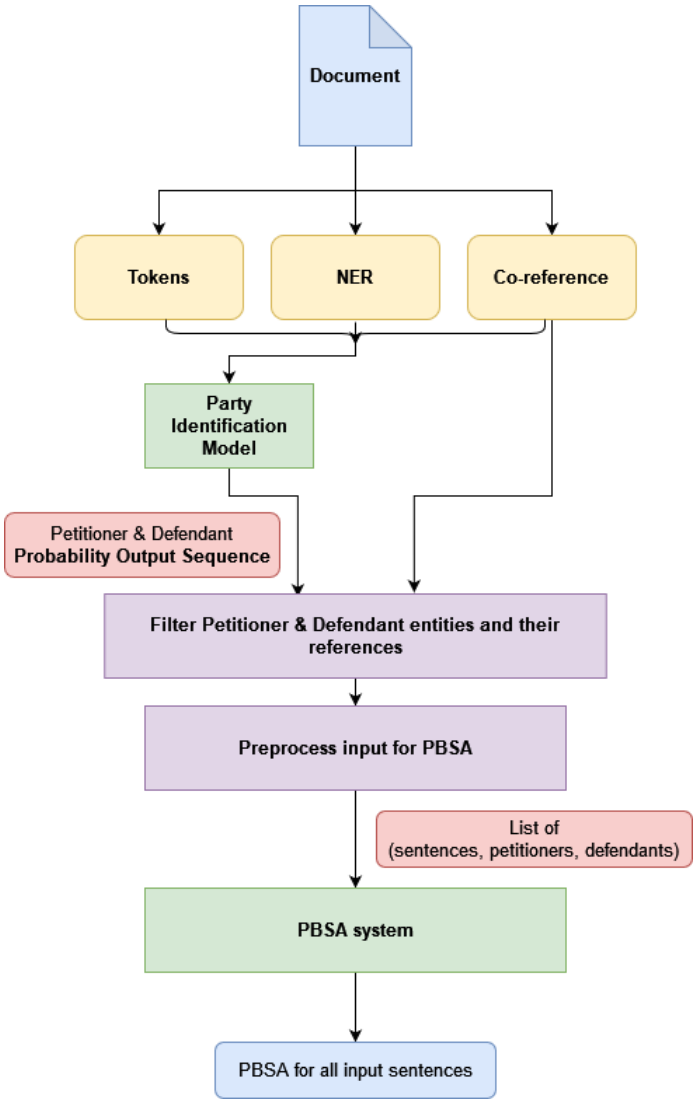

Fig. 7. Pipeline using Petitioner and Defendant Probabilities

3) **nuRefGRU Model:** GRU 512 model trained and evaluated using updated Stanford Co-reference as defined in Section III-F.

TABLE I
Petitioner Metrics

Model	Accuracy	Precision	Recall	F1
Baseline	99.78%	85.03%	82.12%	83.11%
nuRef	99.58%	81.14%	76.70%	77.42%
nuRefGRU	**99.98%**	**88.12%**	**88.12%**	**88.12%**

TABLE II
Defendant Metrics

Model	Accuracy	Precision	Recall	F1
Baseline	99.70%	70.56%	65.40%	67.00%
nuRef	99.55%	64.58%	63.58%	63.14%
nuRefGRU	**99.94%**	**74.75%**	**74.45%**	**74.58%**

V. Conclusion and Future Work

The unified party identification and party based sentiment analysis pipeline reduces a lot of manual work that needs to be done in contrast to using these models separately. Since the combined pipeline outputs the sentiment for each party per sentence for a court case, it can be effectively used to create a sentiment annotated dataset from a large amount of case law document data. The final output of the unified pipeline itself may not represent an end use case, but this dataset will serve as a starting point for further analysis of legal documents using party wise annotated sentiment.

Predicting the winning party of a court case is one such extended usage of the pipeline as it requires the party-wise evaluation of arguments. The derived pipeline can be used to generate a party wise sentiment annotated court case dataset to predict the winning party of a court case which will be an important insight for legal professionals. Likewise there are other use cases for which researchers can fit the pipeline to a bigger architecture according to their task.

While this pipeline reduces manual work, it is important to note that there's room for improvement in both models individually. Identifying such optimizations and fine tuning the models will increase the accuracy of the pipeline overall. Therefore, future work for this pipeline is majorly two fold. Improving the individual model performances can be stated as one important future work. This also includes creating an annotated dataset for evaluation, where the party and the respective sentiments for each party are annotated per each sentence of a given paragraph/document. On the other hand using the pipeline in a bigger architecture for research or practical use case can be stated as another valuable future work.

References

[1] "Legal party," https://www.law.cornell.edu/wex/party, accessed: 2021-05-27.

[2] "Case law," https://www.law.cornell.edu/wex/case law, accessed: 2021-05-27.

[3] M. D. Devika, C. Sunitha, and A. Ganesh, "Sentiment analysis: A comparative study on different approaches," *Procedia Computer Science*, vol. 87, pp. 44–49, 2016, fourth International Conference on Recent Trends in Computer Science & Engineering (ICRTCSE 2016).

[4] Z. Madhoushi, A. Razak, and S. Zainudin, "Aspect-based sentiment analysis methods in recent years," *Asia-Pacific Journal of Information Technology & Multimedia*, vol. 08, 06 2019.

[5] 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," *IEEE International Conference on Industrial and Information Systems (ICIIS)*, pp. 1–6, 2017.

[6] ——, "Legal Document Retrieval using Document Vector Embeddings and Deep Learning," in *Science and Information Conference*. Springer, 2018, pp. 160–175.

[7] 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.

[8] ——, "Semi-Supervised Instance Population of an Ontology using Word Vector Embedding," in *Advances in ICT for Emerging Regions (ICTer), 2017 Seventeenth International Conference on*. IEEE, September 2017, pp. 1–7.

[9] 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.

[10] 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," in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2018, pp. 260–265.

[11] G. Ratnayaka, N. de Silva, A. S. Perera, and R. Pathirana, "Effective approach to develop a sentiment annotator for legal domain in a low resource setting," *arXiv preprint arXiv:2011.00318*, 2020.

[12] I. Rajapaksha, C. R. Mudalige, D. Karunarathna, N. de Silva, G. Rathnayaka, and A. S. Perera, "Rule-based approach for party-based sentiment analysis in legal opinion texts," *arXiv preprint arXiv:2011.05675*, 2020.

[13] C. R. Mudalige, D. Karunarathna, I. Rajapaksha, N. de Silva, G. Ratnayaka, A. S. Perera, and R. Pathirana, "Sigmalaw-absa: Dataset for aspect-based sentiment analysis in legal opinion texts," *arXiv preprint arXiv:2011.06326*, 2020.

[14] I. Rajapaksha, C. R. Mudalige, D. Karunarathna, N. d. Silva, A. S. Perera, and G. 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*. Springer, 2021, pp. 125–137.

[15] 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.

[16] G. Ratnayaka, T. Rupasinghe, N. de Silva, V. Gamage, M. Warushavithana, and A. S. 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.

[17] G. Ratnayaka, T. Rupasinghe, N. de Silva, M. Warushavithana, V. Gamage, M. Perera, and A. S. Perera, "Classifying Sentences in Court Case Transcripts using Discourse and Argumentative Properties," *ICTer*, vol. 12, no. 1, 2019.

[18] C. Samarawickrama, M. de Almeida, N. de Silva, G. Ratnayaka, and A. S. 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)*. IEEE, 2020, pp. 494–499.

[19] M. de Almeida, C. Samarawickrama, N. de Silva, G. Rat- nayaka, and A. S. Perera, "Legal Party Extraction from Legal Opinion Text with Sequence to Sequence Learn- ing," in *2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer)*. IEEE, 2020, pp. 143–148.

[20] M. de Almeida, C. Samarawickrama, N. de Silva, G. Ratnayaka, and S. Perera, "Identifying legal party members from legal opinion documents using natural language processing," in *The 23rd International Conference on Information Integration and Web Intelligence*, ser. iiWAS2021. New York, NY, USA: Association for Computing Machinery, 2022, p. 259–266. [Online]. Available: https://doi.org/10.1145/3487664.3487700

[21] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60.

[22] Court of Appeals, "Cao v. Commonwealth of Puerto Rico," *US, 1st Circuit*, no. No. 07-1394, 2008.

[23] S. Jayasinghe, L. Rambukkanage, A. Silva, N. de Silva, and A. S. Perera, "Party-based sentiment analysis pipeline for the legal domain," in *2021 21st International Conference on Advances in ICT for Emerging Regions (ICter)*. IEEE, 2021, pp. 171–176.

[24] 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.

[25] D. Tang, B. Qin, X. Feng, and T. Liu, "Effective LSTMs for target-dependent sentiment classification," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 3298–3307. [Online]. Available: https://aclanthology.org/C16-1311

[26] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based LSTM for aspect-level sentiment classification," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 606–615. [Online]. Available: https://aclanthology.org/D16-1058

[27] D. Ma, S. Li, X. Zhang, and H. Wang, "Interactive attention networks for aspect-level sentiment classification," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 4068–4074. [Online]. Available: https://doi.org/10.24963/ijcai.2017/568

[28] S. Gu, L. Zhang, Y. Hou, and Y. Song, "A position-aware bidirectional attention network for aspect-level sentiment analysis," in *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 774–784. [Online]. Available: https://aclanthology.org/C18-1066

[29] Q. Liu, H. Zhang, Y. Zeng, Z. Huang, and Z. Wu, "Content attention model for aspect based sentiment analysis," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 1023–1032. [Online]. Available: https://doi.org/10.1145/3178876.3186001

[30] D. Tang, B. Qin, and T. Liu, "Aspect level sentiment classification with deep memory network," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 214–224. [Online]. Available: https://aclanthology.org/D16-1021

[31] P. Chen, Z. Sun, L. Bing, and W. Yang, "Recurrent attention network on memory for aspect sentiment analysis," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 452–461. [Online]. Available: https://aclanthology.org/D17-1047

[32] C. Zhang, Q. Li, and D. Song, "Aspect-based sentiment classification with aspect-specific graph convolutional networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4568–4578. [Online]. Available: https://aclanthology.org/D19-1464

[33] I. Caswell, J. Kreutzer *et al.*, "Quality at a glance: An audit of web-crawled multilingual datasets," *arXiv preprint arXiv:2103.12028*, 2021.

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

[35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding." ACL, Jun. 2019, pp. 4171–4186.

[36] Court of Appeals, "Tobar v. US," *US, 9th Circuit*, vol. 639, no. No. 08-56756, p. 1191, 2011.