SIGMALAW - PREDICTING WINNING PARTY OF A LEGAL CASE USING LEGAL OPINION TEXTS

Sahan Jayasinghe (170264C)

Lakith Rambukkanage (170481M)

Ashan Silva (170584G)

B.Sc. Engineering Honours

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

March 2022

SIGMALAW - PREDICTING WINNING PARTY OF A LEGAL CASE USING LEGAL OPINION TEXTS

Sahan Jayasinghe (170264C)

Lakith Rambukkanage (170481M)

Ashan Silva (170584G)

Dissertation submitted in partial fulfillment of the requirements for the degree
B.Sc. Engineering Honours

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

March 2022

DECLARATION

We declare that this is our own work and this Dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. We retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

Signature: Date:

Signature: Date:

The supervisors should certify the Dissertation with the following declaration.

The above candidate has carried out research for the B.Sc. Engineering Honours Dissertation under our supervision. We confirm that the declaration made above by the student is true and correct.

Name of Supervisor: Dr. Shehan Perera

Signature of the Supervisor: Date:

Name of Supervisor: Dr. Nisansa de Silva

Signature of the Supervisor: Date:

ACKNOWLEDGEMENT

First and foremost, we would like to thank Dr. Adeesha Wijayasiri, Coordinator of Final Year Projects, Department of Computer Science and Engineering, University of Moratuwa, for organizing and coordinating the Final Year Project work for us to get a good learning and research experience.

Next, we would like to extend our sincere gratitude towards our Project Supervisor, Dr. Shehan Perera, Department of Computer Science and Engineering, University of Moratuwa, for providing the opportunity to work on this project and for guiding us throughout the project.

We are much grateful for the constant support and guidance provided to us by Dr. Nisansa de Silva, Department of Computer Science and Engineering, University of Moratuwa, as our external project supervisor. He was instrumental in making this project a success and we would like to acknowledge, with gratitude, the invaluable guidance rendered towards us.

We would also like to take this opportunity to offer our sincere gratitude to Mr. Gathika Ratnayaka for providing constant support and sharing his expertise and experiences on the project with us.

Furthermore, we would like to gratefully acknowledge Ms. Madhavi Perera, for conducting sessions on the legal domain and for the support provided in the domain specific data annotation processes.

We would like to express our gratitude towards Dr. Shantha Fernando and Dr. Charith Chitraranjan for providing direction and valuable feedback through the project evaluation process throughout the project timeline.

Finally we would like to extend our gratitude to all those who assisted, motivated and directed us throughout this project and provided support in numerous ways to complete this research successfully.

ABSTRACT

Natural Language Processing has undergone rapid development over the past years, to cater for the growing needs to analyse huge amounts of text data. Legal domain is such a domain where data is mostly available in the textual format, along with its inherent domain complexities. In this research the aspect of predicting the winning party of a court case using legal opinion texts is explored through four stages. First a party based sentiment analysis pipeline is formulated to identify sentiment of each sentence with respect to the legal parties of each case. Afterwards, a model to identify critical sentence of a court case is trained to be included in the final features. Then a sentence embedding model trained specifically for the legal domain is used to give more visibility to the final model. Finally a winning party prediction model is built by combining these three subsystems. We hope this research work will assist legal professionals to get a benchmark on their case preparation and an initial prediction on the case outcome efficiently.

Keywords: Natural Language Processing, Legal Domain, Deep Learning, Machine Learning

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
ATAE-LSTM	Attention-based LSTM with Aspect Embedding architecture
Bi-RNN	Bidirectional Recurrent Neural Networks
DPL	Defendant probability List
GCN	Graph Convolution Networks
GRU	Gated Recurrent Units
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
NER	Name Entity Recognition
NLP	Natural Language Processing
PBSA	Party-Based Sentiment Analysis
PE	Party Extraction
PoS	Parts of Speech
PPL	Petitioner probability list
RNN	Recurrent Neural Network
STS	Semantic Textual Similarity
SVM	Support Vector Machines
US	United States

CHAPTER 1

INTRODUCTION

Natural Language Processing (NLP) is a rapidly emerging field in research aspects and practical applications. Many NLP techniques such as Name Entity Recognition (NER) , Sentiment Analysis, Word and Sentence Embeddings can be leveraged to automate many tedious tasks across various domains. The advantages of NLP is very effective in fields where large amounts of text data is naturally present. With the use of NLP in such fields, a lot of manual work can be automated and insights can be readily extracted in matter of seconds. The Legal Domain is such a domain where textual data is available in bulk. In addition to that, the legal domain text corpora keeps on growing on a daily basis.

1.1 NLP in the legal domain

The legal domain consists of huge amounts of textual data in the form of constitutions, statues, regulations and case law documents. In addition to having such high volumes of unstructured text data it keeps on growing on a daily basis. As much as the availability of text data enables the use of NLP in the legal domain, what makes it necessary is the inefficiencies and intellectual demand introduced by the abundance of data and domain complexity. Legal professionals have to interact with these documents on a daily basis which are repetitive and requires domain specific skills. Research of NLP in the legal domain could intend to address these concerns, and reduce workloads by assisting legal professionals.

1.2 Case Law

Case Law documents are one of the components which contributes to the growing text data corpora in the legal domain. Case law documents consists of records of past cases with the discussed facts, preserved officially in order to be used as grounds for ongoing cases. The very use of similar cases to assist the ongoing case is why these documents can be identified as very useful in a predictive sense in the context of NLP.

1.3 Legal Party

In a court case two main parties can be identified. The party filing the case is known as petitioner or plaintiff. In criminal cases, an institution on behalf of the government files the case which is referred to as the prosecutor. The party responding to the case is known as defender, respondent or the accused accordingly. Apart from the main two

parties there may be other third parties as well. It is important to note that a party may be an individual, a group of people, an organization or combinations of the above.

1.4 Case decisions

Court case decisions can vary from case to case depending on the type of the case, nature of the court at which the hearing happens, charges or the offense committed and many other factors. In court of appeals, including the supreme court, hearings correspond to cases already heard by lower courts (apart from the high profile cases such as the cases that are related to the constitution). In these cases the losing party has disagreed to the case decision of the lower court and challenges that decision at a higher court. In these cases the supreme court gives decisions such as the following:

- Affirm - Agree to the decision of the lower court
- Reverse - Reverse the decision of the lower court
- Vacate - Set aside or cancel lower court judgment or order
- Dismiss - Refuse to hear the case (Thereby agree to the lower court decision)
- Dissent - Disagree with the majority decision of the court

But in any case the two outcomes that can happen is the petitioner winning or losing (vice versa for the defender). Therefore regardless of the decision it will always favour one party and penalize the other.

1.5 Past Research Work in the Legal Domain

Apart from the few domain characteristics and notions explained before, there are a lot of domain complexities and practices that make legal domain somewhat hard to tackle with regular software engineering. Even the use of feature based machine learning approaches are somewhat insufficient due to the fact that majority of data is in text format. Also, the amount of annotated data is very scarce. Therefore it is important to conduct research to automate legal domain tasks using NLP and deep learning approaches.

Several such noteworthy research work have been carried out such as researches on legal domain sentence discourse relations [3–5], Party-Based Sentiment Analysis (PBSA) in legal opinion texts [6–8], party identification in the legal domain [9–11], legal domain ontology [12, 13] and many other legal domain related researches [14–17].

The main intention of a lawyer as a legal professional, is to prepare as best as possible for the upcoming cases with the victory of the client party in mind. This preparation

requires thorough study of case law documents of former cases and the preparation of similar documents for the ongoing case. This research focuses on the prediction of the winning party of a court case using NLP, which can be used to benchmark the preparation for the ongoing court case based on the output of previously known court cases.

CHAPTER 2

PROBLEM STATEMENT

Legal domain with its inherent nature, has a lot of documents which are used for various purposes by the professionals in the legal domain. Legal opinion texts are the documents which consist of information about the court cases and they contain valuable information which is/was required to make the final judgment of the court cases. Legal professionals, for example, lawyers, have to spend enormous amounts of time trying to go through legal opinion texts and decide which kind of decision would be the outcome of this court case and whether they have enough facts or arguments to support their claims. Almost all of these tasks would be done manually and will require professional skill, significant effort and as mentioned before, a lot of time. Therefore it would be a valuable contribution towards the legal domain if these tasks could be automated.

Hence the core problem statement for this project is addressing the need of a proper method to predict the winning party of a court case by legal opinion texts, to provide with a measure of confidence of a case decision and to act as a benchmark on case preparations for legal professionals.

CHAPTER 3

RESEARCH OBJECTIVES

The objective of the research is to design, develop and validate a methodology to give legal opinion texts as inputs and output the winning party, in other words the case decision.

- **Combine the outputs of "Sigmalaw-absa: Dataset for aspect-based sentiment analysis in legal opinion text" [8], "Identifying legal party members from legal opinion texts using natural language processing" [10] into a single workflow**

  By combining the steps of identifying parties and assigning sentiment scores to the respective parties we can ideally reduce a lot of manual work. This pipeline will then be capable of predicting the sentiment towards each party member mentioned in a court case per sentence. Using the pipeline output for sentences in a court case will hopefully assist to calculate the impact towards wining or losing a case with respect to each major party.

- **Develop a system for predicting the criticality of a case sentence with respect to the case decision and impact towards the parties**

  US Supreme Court case documents consists of a large number of sentences which reflects case background, evidence, previous court decisions and reasoning. To make the case content analysis easy for legal officials, we plan to develop a system that can predict the criticality of case sentences with respect to the final decision and impact towards the petitioner/defendant party.

- **Design a sentence embedding specifically for legal domain**

  Even though there are many general-purpose sentence embedding models pretrained on large text datasets such as Wikipedia book corpus available online, we plan to train a domain-specific sentence embedding using the legal text corpus that can be extracted from thousands of United States (US) Supreme Court cases. With the use of the embedding system, hopefully we will be able to obtain better representations for the case sentences rather than using general-purpose models.

- **Explore and develop a methodology to predict the winning party of a court case using the prepared data set.**

  Primary objective here is to use different NLP techniques to design develop and validate a methodology to predict the winning party of a given case. The model

would take court cases as inputs and output the court case decision thereby the winning party can be identified as per the favoured party of the output decision.

CHAPTER 4

LITERATURE REVIEW

4.1 Winning Party Prediction

The researches Shaikha et al. [18] have formulated three categories to classify the past approaches to predict the outcome of a legal case. The three approaches use political or social science based, linguistics based or legal domain based features as the descriptors for the machine learning algorithms they use. They have defined 19 features with respect to legal domain, that would potentially impact the decision of a criminal court case. Feature extraction have to be manually done by going through a court case and the person reading the case should have expertise of some level to identify the features. After the feature extraction and preprocessing, they have conducted classification under eight different algorithms such as Regression Trees, Bagging and Random Forests, Support Vector Machines and K-nearest neighbours. They have achieved best performance from Classification and Regression Trees.

Research done by Aletras et al. [19] on predicting the decision of the European court of human rights, claims it as the first systematic approach to predicting winning parties with the usage of NLP. They have approached the problem as a binary classification problem using Support Vector Machines and N-grams and topics as features.

The work of Liu and Chen [20] also proposes a classification approach for identifying the winning party of a court case. The system is configured as two phases where in the 1st phase, the Article Classification model extracts top k articles that are cited in the case document, and in the 2nd phase, the Judgement Classification model predicts the judgement that is more likely to be given in the court case. As features they have considered domain specific aspects such as punishment, cited statutes and NLP specific features such as sentiment. The process has two phases. The first phase identifies related statutes to the court case and vectorize them. The second phase calculates the sentiment score for the input court case and is combined with the related statute vector which is passed on to the classification model to get the prediction of the judgment.

In the research conducted by Katz et al. [21], a tree based approach which employs new feature engineering techniques is proposed. The dataset used for the research is from the United States Supreme Court and they have considered the impact from political biases for the decisions as well. They have considered the prescriptive capabilities of the model and therefore have used data ranging over multiple presidential terms to generalize more better. They have used features already present with there chosen dataset and some have been introduced by them. Out of the 7700 cases used, they have achieved 69.7% accuracy and individual judge votes with 70.9% accuracy.

Lage-Freitas et al. [22] propose a machine learning approach to develop a system

that predicts Brazilian court decisions, which may be used as a supporting tool or a benchmark by legal professionals. They have designed the approach to calculate both the decision class and the unanimity of decisions and achieved significant accuracy.

4.2 Party Identification in the Legal Domain

Samarawickrama et al. [9] have studied the entity occurring frequency at as the subject of a sentence to extract the parties. They have used two models for this purpose: 1) NER and co-reference resolution to identify entities, and 2) Calculate a probability output based on subject frequency. Final probability output is subjected to a threshold to identify the entities belonging to a party. In contrast, in the study by de Almeida et al. [11], first an NER Model is used for identifying people and organizations, on which a mask is applied. Followed by that co-reference resolution is used to resolve make representations uniform for the same entity. An 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 is used to decide whether each word belongs to a party or not.

In their recent study [10] have introduced a novel method to identify parties more accurately using deep learning. A number of deep learning models have been trained and evaluated while experimenting with different architectures of Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) . A data set (1000 paragraphs) created using court case documents of US Supreme Court published by Sugathadasa et al. [14] is used for training. The process consists of the following steps: 1) Tokenizing, 2) Embedding, 3) Masking, and 4) using the Neural Network. In the *Tokenizing* step NER and co-reference resolution are used to identify entities. As the *Embedding* a Word2Vec [23] model of 300 dimensions[1] is used. The *Masking* step uses the identified entities which 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.

4.3 Party Based Sentiment Analysis (PBSA)

Sentence level sentiment is used to calculate sentiment for each party in the study by Rajapaksha et al. [6]. To allocate the sentiment to each party they have used a simple convention of allocating the sentiment to the party mentioned in the phrase. Stanford Core NLP [24] co-reference resolution is used to resolve for pronouns and other references of the same entity. Further, they have used the constituency parser to breakdown the phrases for sentiment annotation.

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

In the work of Mudalige et al. [7] major drawback in NLP related to the legal domain is addressed which is the lack of proper datasets. They have created a publicly available dataset for Aspect (Party) Based Sentiment Analysis. A dataset originally extracted from the United States Supreme Court consisting of nearly 2000 sentences is annotated. Two main parties of a case which are petitioner and defendant, and sentiment for each party within the sentences are labeled as positive (+1), neutral (0) and negative (-1), overall sentiment of the sentence regardless of the parties labeled as positive (+1), neutral (0) and negative (-1).

4.4 Cross Entropy Loss

As explained in the work of Zhang and Sabuncu [25], classification using Mean Absolute Error (MAE) as the loss function works only under few assumptions. When the classification labels of the dataset becomes noisy MAE starts to work poorly, where as Cross Entropy Loss or tuning of Cross entropy loss works well with noisy data. Zhang and Sabuncu [25] have experimented modified Cross Entropy loss functions with synthetically added noise for 3 datasets at different noise levels and all of them have outperformed MAE.

4.5 Critical Sentence identification

In the work of Glaser et al. [26] the classification of sentences in legal contracts based on 9 different semantic classes are explored. The research experiment has been conducted for legal texts in German language. First the Machine Learning (ML) models have been trained for sentences extracted from the German Civil code and later used on legal contracts to extract the corresponding critical sentences in a contract.

Researchers Jagadeesh et al. [27] have proposed a sequential process to summarize documents and then extract important sentences. For this purpose, in the first part they have employed syntactic parsing with Named Entity Recognition (NER) and Parts of Speech (PoS) Tagging followed by feature extraction based on the NER, PoS tags, word frequencies etc. Afterwards sentences are scored based on the features and ranked accordingly. The system allows to extract sentences based on a query by generating a coherence score is a key element.

Hirao et al. [28] have explored the sentence extraction aspect by using Support Vector Machines (SVM) . The approach to extracting the critical sentences from a document is similar to text summarizing, and classifying the sentences based on two classes of important and unimportant. They have been able to beat three other approaches for text summarizing at the time as well.

CHAPTER 5

METHODOLOGY

5.1 Party-based Sentiment Analysis Pipeline for the Legal Domain

This section covers the research work carried out in [29]. The objective of this study is to define a pipeline which uses the output of the Party Extraction (PE) system of Samarawickrama et al. [10] (Referred here onward as the *PE*) to generate the input for sentence-wise Party-based Sentiment Analysis system of Mudalige et al. [7] (Referred here onward as the *PBSA*). To achieve this, we came up with a more accurate solution for party identification through algorithmic improvements and the adapter implementation for connecting the two sub-systems [7, 10]. The evaluation results presented in the upcoming section, compare the improved version vs. the previous work of PE models.

5.1.1 Party Extraction System

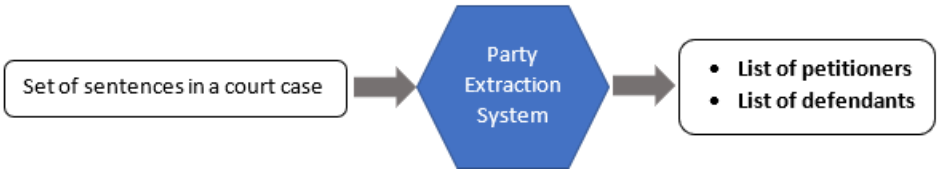

Fig. 5.1: Party Extraction System

Samarawickrama et al. [10] have trained a sequence-to-sequence model to identify legal party members mentioned in the input text using pre-trained Word2Vec model of 300-dimensional for word embeddings, and Stanford NER and Co-Reference annotations to capture real-world entities and their references. Their best performing model is the GRU model with 512 output units (referred as *GRU model* in this report from 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 5.1, this model serves as the *Party Extraction Sub-System* of our pipeline.

5.1.2 Party-based Sentiment Analysis (PBSA) System

Mudalige et al. [7] have created a public dataset for Party-Based Sentiment Analysis which consists of sentences taken from the US Supreme Court cases dataset released by Sugathadasa et al. [14]. PBSA dataset labels each sentence with mentioned party members and their respective sentiments. They have concluded after their experiments that the ensemble model implemented using BERT [30] embeddings, Attention-based

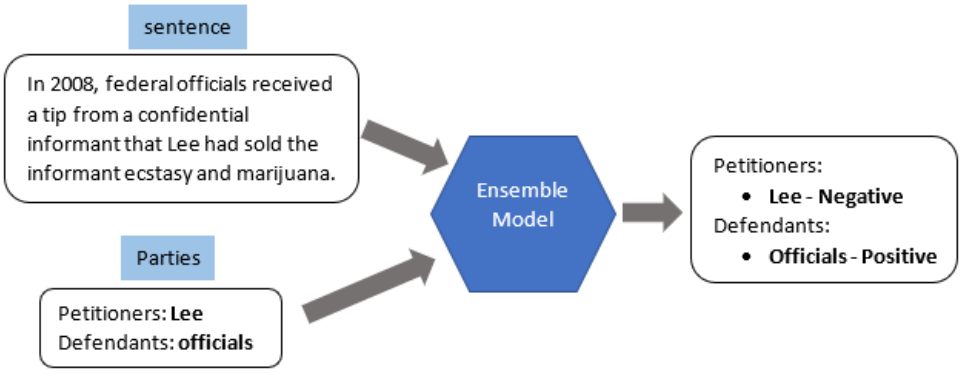

Fig. 5.2: PBSA System working on a sentence from Lee v. United States [1]

LSTM with Aspect Embedding architecture (ATAE-LSTM) [31], 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 *PE*. As shown in Fig 5.2, this model serves as the *PBSA System* of our pipeline.

5.1.3 Pipeline Overview

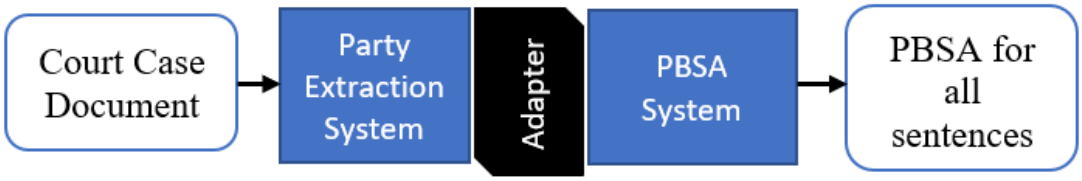

Fig. 5.3: Pipeline Overview

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. 5.3. *PE* output is passed on to the Adapter which generates the input to *PBSA*. Adapter outputs the sentences one-by-one along with the mentioned party members. Petitioner and defendant entities and their references in each sentence of the given text ordered exactly as they appear to match the input format of *PBSA*.

5.1.4 Baseline Model

As the starting point of implementing the pipeline, we straightforwardly used the *PE* defined by Samarawickrama et al. [10] which consists of a seq-to-seq 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. However, for certain scenarios, the extracted names of party members

deviated from expected string due to a description tailing behind the actual name of the entity. An example of this faulty co-reference annotation is shown in Fig. 5.4.

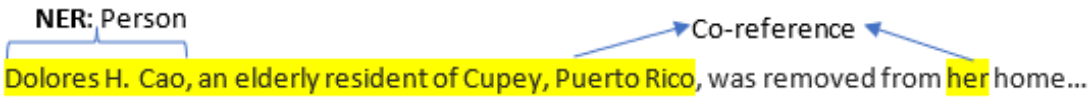

Fig. 5.4: Erroneous co-reference in *Cao v. Commonwealth of Puerto Rico* [2]

In Fig. 5.4, the pronoun *her* should be identified as the co-reference of *Dolores H. Cao*. But instead, the extraction function of Samarawickrama et al. [9] forms an erroneous entity where the tailing description of *Dolores H. Cao* is also included in the entity.

5.1.5 nuRef model: Improving Stanford Co-Reference Output

To mitigate the issue of extracting the entities with a tailing description, we implemented Algorithm 1 to update the co-referenced entities to actual entity name. Since the goal is to remove descriptive fields, the execution goes through each co-reference entity in the annotation and incorporates Stanford NER to extract out the actual words representing the entity.

Fig. 5.5 shows the result of sending the faulty entity tagged in Fig. 5.4 through the *Co-Ref update* Algorithm. 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. 5.5, it is *Dolores H. Cao*.

We define the *nuRef* model with the GRU of Samarawickrama et al. [10] which was trained using masked vectors generated using Stanford Co-Reference annotation and the updated co-reference. When the *nuRef* was tested, it was observed that even though the entity strings 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 5.1.4. Therefore, it was decided that the GRU model should be re-trained using the masked vectors generated using the updated Co-Reference.

5.1.6 nuRefGRU model: Training the GRU using the Updated Co-Reference

We followed the same approach proposed by Samarawickrama et al. [9] for generating mask values for entities and their references. However, while the original model uses

Algorithm 1: Co-Reference Update Algorithm

Input: TokenList and Stanford Co-reference annotation
Output: Updated Co-Reference annotation

```
Function UpdateCoRef(TokenList, CoRefList):
    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 := FindSameNERWords(CurrentToken.NER,
                  EntityTokens[CurrentTokenIndex : Entity.EndTokenIndex])
                CoRef.Entity.text := SameNERWords
    return CoRefList
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
```

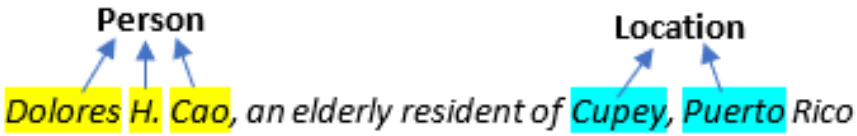

Fig. 5.5: Co-Reference update using NER (Cao v. CoPR [2])

Stanford NER fed to Stanford Co-Reference, we fed Stanford NER to the updated Co-Reference.

We define this model with the GRU retrained with updated co-refernce as *nuRefGRU* model. It was observed that the *nuRefGRU* model out-performs not only the *nuRef* model but also the *Baseline* model. Therefore, with the *nuRefGRU* model we proceeded to the implementation of the adapter for generating the input for the *PBSA* system[7].

5.1.7 Pipeline Implementation 1

The inputs for the *adapter* set between the *PE* and *PBSA* are: list of tokens, updated co-references, and the outputs of the *PE*. 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* [7].

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

```
1  Function AdapterCoref(TokenList, CoRefList, PetitionersList,
     DefendantsList):
2      LegalEntityPositions := Object()
3      for each CoRef in CorefList do
4          if CoRef.Entity in PetitionersList then
5              for each Reference of CoRef.Entity do
6                  sentenceIndex := Reference.sentInd
7                  LegalEntityPositions.sentenceIndex.Petitioner := {TokenIndex:
                     CoRef.text}
8              remove Coref.Entity from PetitionersList
9          else if Coref.Entity in DefendantsList then
10             for each Reference of CoRef.Entity do
11                 sentenceIndex := Reference.sentInd
12                 LegalEntityPositions.sentenceIndex.Defendant := {TokenIndex:
                     CoRef.text}
13             remove Coref.Entity from DefendantsList
14     if PetitionersList is not empty then
15         for each Petitoner in PetitionersList do
16             find location of Petitioner in TokensList and update LegalEntityPositions
17     if DefendantsList is not empty then
18         for each Defendant in DefendantsList do
19             find location of Defendant in TokensList and update LegalEntityPositions
20     return LegalEntityPositions
21 end Function
```

Algorithm 2 is defined to output the petitioner and defendant entities in the same order they are mentioned in each sentence for PBSA System. The algorithm 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.

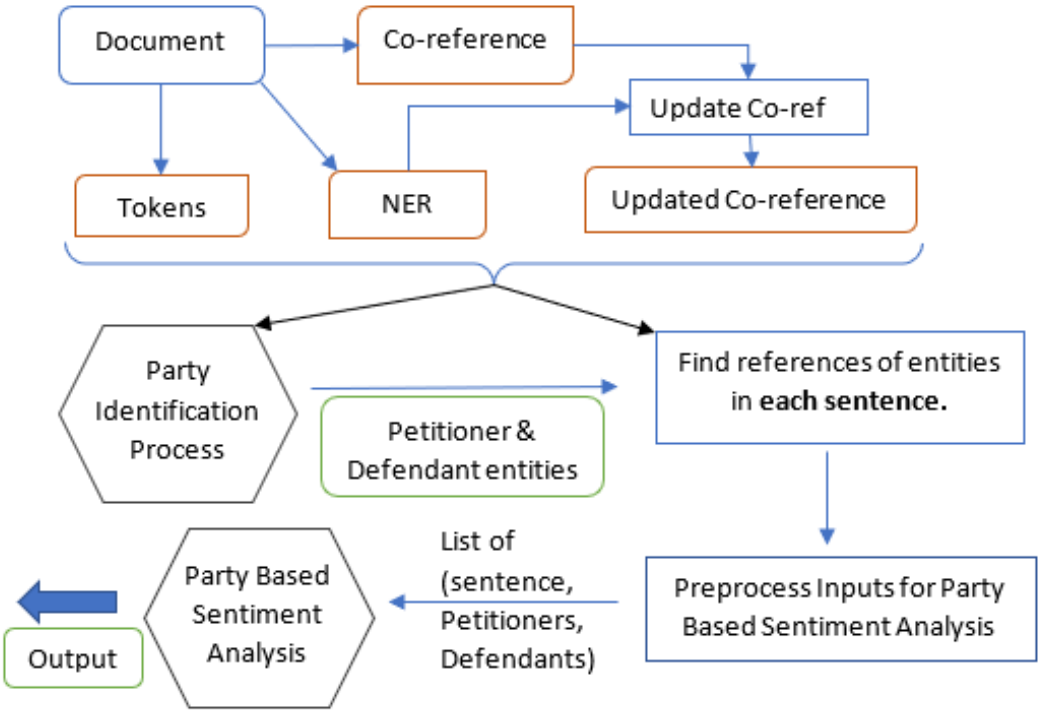

Fig. 5.6: Pipeline with updated Co-Reference Annotation

Workflow of the Pipeline Implementation 1 is elaborated in Fig. 5.6.

5.1.8 Drawbacks of Implementation 1

Implementation 1 (Section 5.1.7) depends on both Stanford Annotation accuracy and the deep learning model for party probability prediction accuracy. The final algorithm of *PE* defined by Samarawickrama et al. [9] 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 issue, the *PE* 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*).

Ex.2 - Tobar v. US [32]

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. Plaintiffs then sued the United States for damages resulting from these events.

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*.

Since there's both positive and negative aspects of the Implementation 1 (Section 5.1.7), we researched for a different approach to overcome the existing issues.

5.1.9 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 PBSA system [7].

Algorithm 3 is defined 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 decide a token is either petitioner or defendant. When the token has the petitioner probability value greater than or equal to the threshold and the defendant probability 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 are neglected.

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

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
```

we provide for the algorithm 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 6.1.

The workflow for the combined systems along with the adapter we defined in this approach is elaborated in Fig. 5.7.

With this implementation of the pipeline, the main issue we faced in the Implementation 1 (Section 5.1.7) is mitigated, since the identification of petitioner and defendant

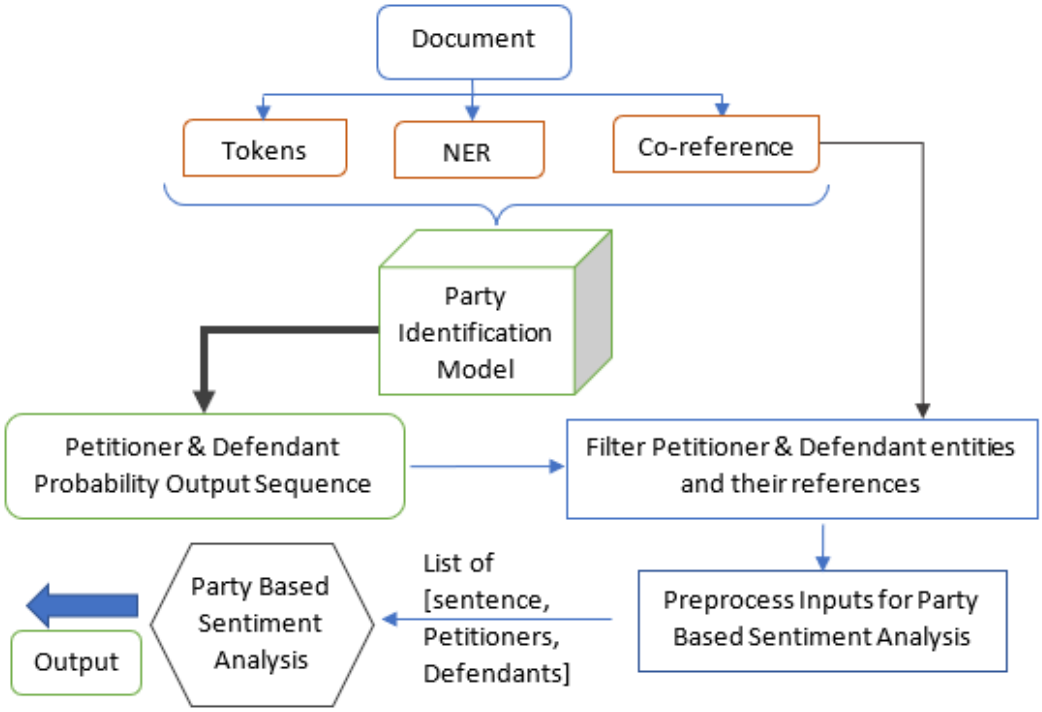

Fig. 5.7: Pipeline using Petitioner and Defendant Probabilities

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*, since there is no co-reference identification unlike in the Implementation 1 (Section 5.1.7).

5.2 Critical Sentence Identification in Legal Cases

This research is carried out to achieve the goal of identifying critical arguments made in a court case by each party which in turn leads to a higher impact on the case decision [33]. The dataset for this research is an extension of the Party Based Sentiment Analysis (PBSA) dataset created by Mudalige et al. [7] which is publicly available at OSF[1]. The PBSA dataset consists of 1822 exact sentences and meaningful sub sentences extracted from 25 case law documents of United States Supreme Court by Sugathadasa et al. [14], along with the sentiment for each party present in them and the overall sentiment.

[1]SigmaLaw ABSA Dataset: https://osf.io/37gkh/

5.2.1 Dataset Preparation

We used 1822 sentences of the PBSA dataset developed by Mudalige et al. [7] along with the party and sentiment annotations. The dataset was extended by adding the decision of each court case for each sentence. We identified four classes that a sentence can be belonged to, based on the winning party of the court case and the impact of the sentence towards the party. A sentence can be classified as follows.

1. Petitioner lose and has negative impact towards petitioner (Petitioner lose and negative)
2. Petitioner lose and has positive impact towards petitioner (Petitioner lose and positive)
3. Petitioner win and has negative impact towards petitioner (Petitioner win and negative)
4. Petitioner win and has positive impact towards petitioner (Petitioner win and positive)

For simplicity, we will refer the above categories in a shorter form as mentioned in the brackets. According to above classes, a sentence can be considered as critical when it has,

- A positive impact towards petitioner in a case document where petitioner won
- A negative impact toward petitioner in a case document where petitioner lost

In PBSA dataset, the parties mentioned in each sentence are annotated with their sentiment labels. We used those sentiment labels to calculate the impact towards petitioner for each sentence.

Jae Lee V. US

Case Sentence 1 from Jae Lee V. US [1]

After obtaining a warrant, the officials searched Lee's house, where they found drugs, cash, and a loaded rifle.

In the case sentence 1, the petitioner is Lee and the defendant party is represented by the officials. PBSA dataset contains the sentiment labels for the words *Lee*, *officials* and *they* as follows.

- Petitioners: Lee – Negative (-1)
- Defendants: Officials – Positive (+1), they - Positive (+1)

According to the sentiment labels, the impact towards petitioner in the sentence **??** can be calculated as negative by subtracting average sentiment for defendant members from average sentiment of petitioner members.

Whenever there is a sentence which mentions only the members of the defendant party, we consider the inverse sentiment of the defendants as the impact towards petitioner for the purpose of categorizing the sentence according to the 4 classes. Within the 1822 sentences, there were 214 sentences which reflected a neutral sentiment towards any of the parties. We removed those sentences since those are insignificant for the sentence importance prediction and also they would lead to a high class imbalance due to the abundance. In case of providing a sentence with a neutral sentiment to the model trained on the following classes, the output probabilities should not significantly be bias to any class.

Remaining 1608 sentences are labeled as Petitioner win/lose regarding the decision of the case document the respective sentence belongs to. Sentence counts of each class are:

- Petitioner lose and negative – 226
- Petitioner lose and positive – 230
- Petitioner win and negative – 687
- Petitioner win and positive – 465

Since most of the US Supreme court cases are appeals by petitioners, there is a marginally higher number of sentences which affect negatively towards petitioner.

5.2.2 Multi-class Classification Model

The proposed architecture for the multi-class classification task consists of 3 components: Token Embedding model, a Mean Pooling Layer and a Dense Layer. For the tokenization process, we used pretrained BertTokenizer containing vocabulary of 30000 words and sub-words. For the Embedding model, we used Bert-base-cased model [2] pretrained on Wikipedia text. Final Dense layer consists of nodes equal to number of classes.

Bert Tokenizer is configured to generate tokens for the words and sub-words identified in the given input text and create a padded token sequence of 128 long. An input mask is also generated so that the Bert model can distinguish the tokens representing the text. Bert model generates a 768-dimensional embedding for a single token. To obtain a unified representation for the input text sequence, there are 3 techniques mentioned by Reimers and Gurevych [34]: using the [CLS] token embedding, taking the

[2]bert-base-cased: https://huggingface.co/bert-base-cased

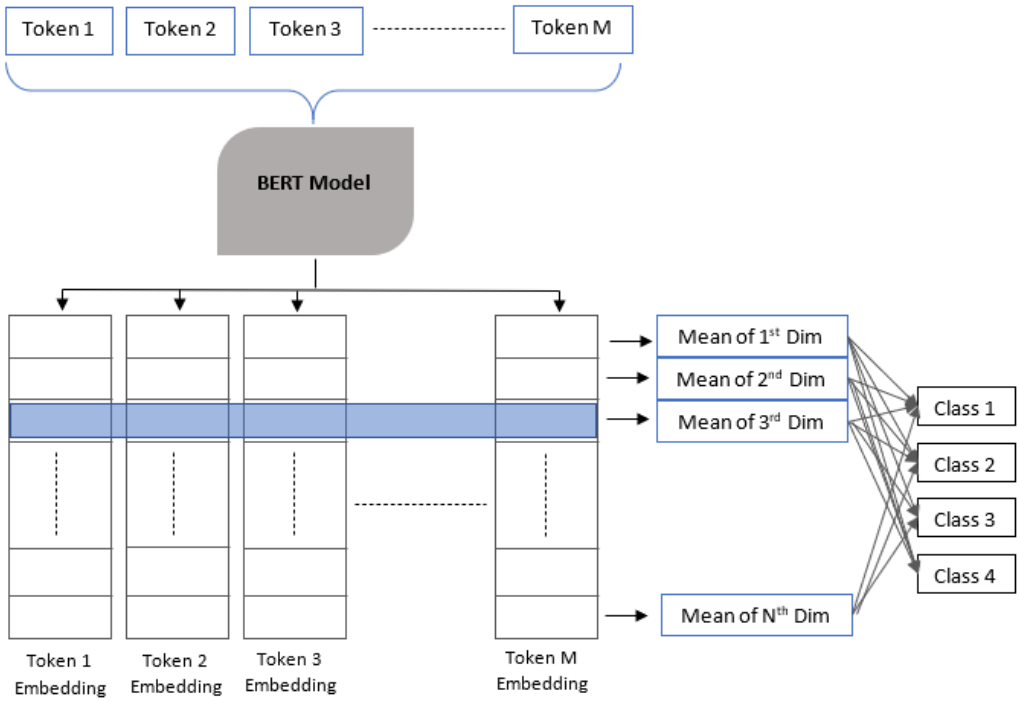

Fig. 5.8: Model Architecture

mean of token embeddings at each dimension (Mean Pooling), taking maximum of the token embeddings at each dimension (Max Pooling). In this research, we used Mean Pooling to obtain a generalized representation for the sentences provided to the model as input.

5.2.3 Task-specific Loss Function

Categorical Cross Entropy loss is the most promising loss function for multi-class classification tasks. Cross Entropy loss is calculated using only the probability of the labeled class. But we defined a new loss function which takes into account the probabilities of rest of the classes instead of the labeled class.

The classification task discussed in this section contains four classes that can be separated into two polarities depending on the decision of the court case.

TABLE 5.1: CLASS POLARITY

Petitioner lose and negative	Petitioner win and negative
Petitioner lose and positive	Petitioner win and positive

We defined the loss function as in Algorithm 4, which takes into account the polarity of classes.

Algorithm 4: Loss Function for case sentence classification

```
Function OppositeClassWeightedLoss(ModelOutput, Label, OppositeClassWeight):
    NumClasses := Length(ModelOutput)
    MidClassIndex := (NumClasses + 1) div 2
    SoftmaxOutput := Softmax(ModelOutput)
    Loss := 0
    for ClassIndex:= 0 : NumClasses-1 do
        if Label = ClassIndex then
            Weight := 0
        else if Label < MidClassIndex then
            if ClassIndex ≥ MidClassIndex then
                Weight := OppositeClassWeight
            else
                Weight := 1
        else
            if ClassIndex < MidClassIndex then
                Weight := OppositeClassWeight
            else
                Weight := 1
        Loss := Loss + Weight * log(1 - SoftmaxOutput[ClassIndex])
    return Loss
end Function
```

The goal of the loss function is to penalize more on the probabilities of opposite classes. For an example, when the label is Petitioner lose and negative, the opposite classes are Petitioner win and negative, Petitioner win and positive. In the context of legal domain, predicting a sentence to be in a case document where petitioner lost, while that sentence exists in a case where petitioner won is a severe error when compared to predicting the wrong sentiment (impact).

Algorithm 4 takes in the output of each node at the last layer of the model, then Softmax, shown in equation 5.1, is applied to obtain probabilities of each class, where x_i is the output of the ith node.

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{5.1}$$

MidClassIndex field represents the class index which switches the polarity. According to above four classes, class indices 0 and 1 represents the petitioner lose polarity, class indices 2 and 3 represents the petitioner win polarity. Depending on the

label of the case sentence, a weight is dynamically applied to the probability loss of the opposite classes. Weight for the labeled class is 0. Weight for the classes which has the same polarity as the labeled class is 1.

The weight applied for opposite classes is configured as a hyper-parameter which we have experimented in Section 6.2. When the predicted probabilities of the opposite classes are high, the loss becomes a higher value as shown in Equation 5.2, where C_i is the ith class, L_{c_i} is the loss calculated for the ith class, W_i is the current weight associated with the ith class, and P_{c_i} is the predicted probability of the ith class.

$$L_{c_i} = W_{c_i} * log(1 - P_{c_i}) \tag{5.2}$$

Total loss for an input sentence is the accumulative loss of each class *i*.

5.3 Domain Specific Sentence Embedding

5.3.1 Legal Case Dataset

The dataset used for the training of the embedding was mined from the United States Supreme Court Case Law records found at FindLaw website[3]. Criminal Cases ranging from the year 2000 to 2010 were chosen in order to build the dataset. This specific time period is chosen due to unavailability of full transcripts on recent cases and the unstructured format in older case documents.

After analyzing the dataset, we found out erroneous phrases and redundant parts that need to be fixed to improve the overall quality. We decided to remove text within rounded brackets since they contained redundant information such as previous case citations or semantically unmatched context with respect to the sentence.

There were some abbreviations in legal domain which caused parts of sentences to be split from Stanford tokenizer. We replaced those with their long forms. Two examples are Federal Rule of Criminal Procedure which is mentioned as Fed.R.Crim.P. and Federal Rule of Evidence mentioned as Fed.R.Evid in the documents.

Due to web page styling and emphasis techniques, there were square brackets around letters and words within sentences as shown in the examples. We fixed those cases as well. Also we decided to replace case citations with a keyword [CITE] and remove sentences which contained larger portion of citations. These steps can be summarized as follows.

1. Removed text within rounded brackets.

2. Replaced abbreviations specific to legal domain with their long form.

    - Fed.R.Crim.P. – Federal Rule of Criminal Procedure

[3] https://caselaw.findlaw.com/

- Fed.R.Evid. – Federal Rule of Evidence

3. Removed square brackets around letters or words. (Ex: [T]he, Extend[ed], [petitioner])

4. Replaced citations of previous cases with [CITE] keyword.

5. Removed sentences with more than 25% of [CITE] keyword with respect to all words.

6. Removed numbering from topic sentences (Ex: II., A., 3.)

5.3.2 Auto Encoder

Due to the availability of a large in-domain text dataset, we searched for an unsupervised approach for learning sentence embeddings for the legal domain. We came up with the Auto-Encoder architecture, inspired by the application of Encoder-Decoder architecture in Neural Machine Translation systems [35, 36]. The objective of the Auto-Encoder is to re-construct the original sentence token-by-token in an iterative manner using the state from previous tokens of the sentence and the vector representation for the whole sentence generated by the Encoder.

The workflow of the Auto-Encoder for a sentence containing m tokens at the $(k\text{-}1)^{th}$ iteration of the decoder is displayed in Figure 5.9. Upper section of the diagram represents the Encoder and lower section, the Decoder. The Embedding layers used in both Encoder and Decoder share same embedding matrix populated with pre-trained word embeddings.

Encoder takes in a sentence as a sequence of tokens and outputs a vector representation for the sentence. According to Figure 5.9, Embedding layer outputs a sequence of m vectors which is then passed on to a Recurrent layer with a specified number of units. The final state vector of the Recurrent layer is considered as the sentence embedding which is passed on to the decoder.

Each sentence is padded from the beginning with a *[START]* token and a *[END]* token to mark the beginning and the end of a sentence. Decoder iteratively predicts the next token starting from the *[START]* token at the first iteration to predict the token after the *[START]* token. Figure 5.9 depicts the $(k\text{-}1)^{th}$ iteration of the decoder, where the vector for $(k+1)^{th}$ token is predicted. Decoder takes in the k tokens preceding the $(k+1)^{th}$ token and passes them to the Embedding layer which outputs a sequence of k vectors. These vectors are passes on to a Recurrent layer where the final state vector is concatenated with the sentence vector provided by the Encoder. These concatenated output is passed on to a Dense layer which outputs a vector with the same dimension of the pre-trained embeddings.

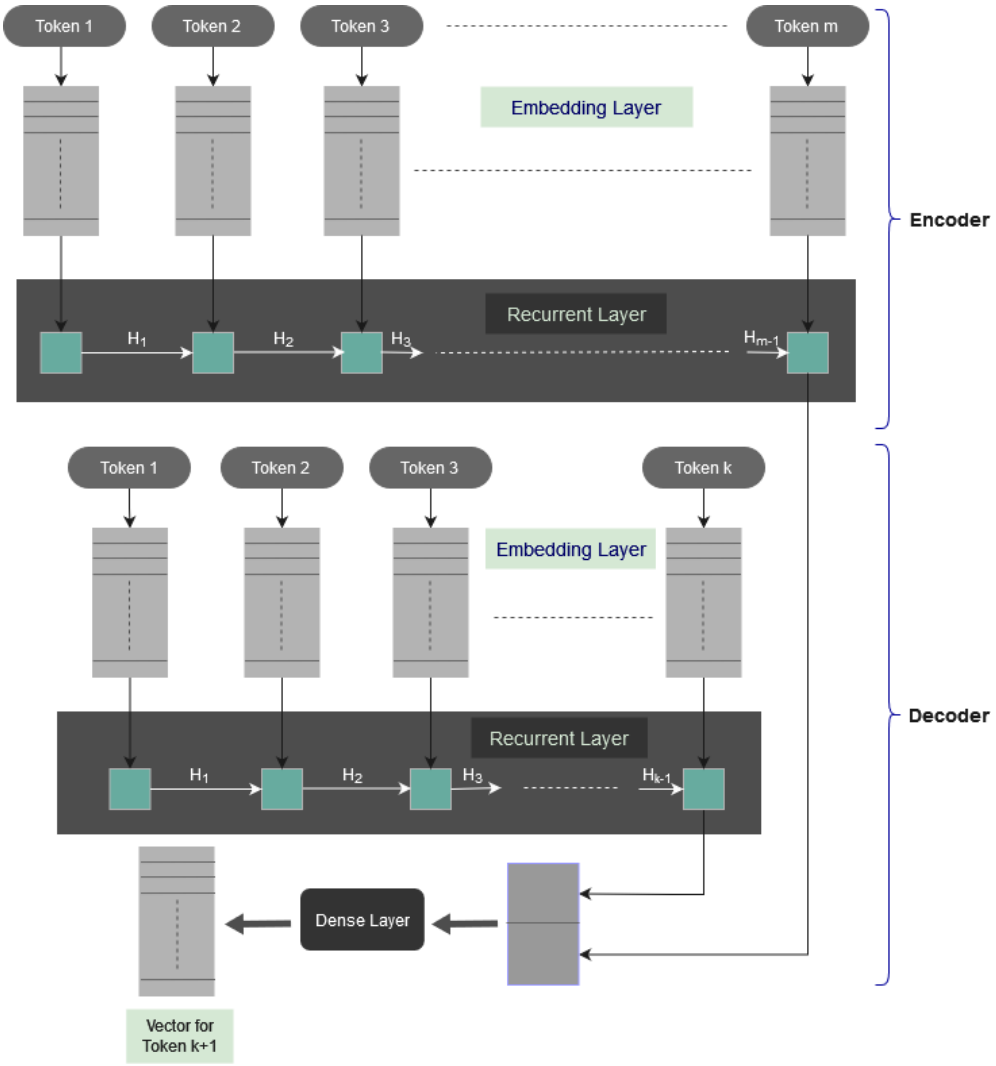

Fig. 5.9: Auto Encoder Architecture

Training loss is calculated at each decoding iteration, using the mean squared error between the predicted token vector and the actual vector obtained from pre-trained word embeddings. Cosine similarity is used as the accuracy metric to evaluate the similarity between predicted and pre-trained word vectors.

The ability to predict the next token at each decoding step is based on the semantic meaning captured by the Encoder for the complete sentence and the state captured by the Decoder's Recurrent layer about the tokens preceding the to-be-predicted token of the sequence.

Word embeddings: Glove[37], Word2Vec[23], FastText[38] are used in the experiments and the results are included in section 6.3.

5.3.3 STS Dataset for Legal Domain

Semantic Textual Similarity (STS) is a measurement used to assess the closeness of two text content with respect to their semantic meaning. Conneau and Kiela [39] defined an evaluation tool kit for universal sentence representation which makes use of STS dataset. STS Benchmark dataset [40] consists of pairs of sentences and the corresponding STS Scores manually annotated for each pair of sentences. STS score is a value between 0 and 5 where the perfect semantic similarity between two sentences is represented by the score of 5. Scores close to 5 represents the sentence pairs that are somewhat producing the same meaning while scores close to 0 represents irrelevant sentence pairs.

This dataset is used for training state-of-the-art sentence embeddings by Reimers and Gurevych [41] through supervised learning approaches. They have shown that sentence embeddings trained using supervised learning perform significantly better in semantic text comparison tasks compared to sentence embeddings trained using unsupervised or self-supervised methods. The supervised training approach used by Reimers and Gurevych [41] optimizes the model based on the difference between the cosine similarity of a sentence pair and the normalized STS score (within the range 0 to 1). The goal of this optimization is to move the vectors representing semantically similar sentences close in the high-dimensional vector space. Moreover, correlation results generated by evaluating models for STS Benchmark dataset is used in comparing the performance of sentence embeddings in general [41, 42].

Understanding the need of a labeled dataset for legal domain to train and evaluate sentence embeddings. With the help of a professional in legal domain, a STS dataset is prepared using sentences taken from US Supreme Court criminal cases. Sample sentence pairs taken from the prepared dataset is displayed in Table 5.2.

TABLE 5.2: STS Legal Dataset - Samples

Sentence 1	**Sentence 2**	**Score**
Petitioner explained that his actions were taken in self-defense.	During the court proceedings, plaintiff argued he was only trying to save himself.	4.25
He did not present any evidence.	There were no evidence to support him.	3.25
The memorandum argued that plaintiff was not a risk to public safety and that he had accepted responsibility for his crime.	Court hearing raised concerns about the public safety.	1.5

First pair of sentences in Table 5.2 has a high STS score because the semantic meaning is same despite some content before the second sentence. Second pair of sentences has somewhat lower score since the first sentence doesn't elaborate the need

for the petitioner to present evidence. It could be about supporting himself or against the opponent party. Last sentence pair has a low score since they are irrelevant despite the mention of public safety.

5.3.4 Multi-task Model for learning Sentence Embeddings

Since we have prepared a large dataset of case sentences as described in Section 5.3.1 and we have obtained a labeled dataset of STS score annotated sentence pairs, we focused on training a model for multiple tasks. To make use of the large set of unlabeled sentences, we defined a task to determine whether a sentence is distorted or not. Legal STS dataset is used for the task of predicting the similarity between two sentences and evaluating against the STS score. We list down the two tasks that the model is trained for:

- Noise added sentence discrimination
- Semantic similarity between a sentence pair

Noise addition process for sentences is done using a random word replacement algorithm. First, a set of general english words is extracted from the case sentence dataset. This set of words does not contain any person names, organization names or punctuation marks. Total number of general words accounts for 17796.

This set of words is used to replace 20% of words within each sentence by picking randomly. With this word replacement, the semantic meaning of the sentence is distorted. An example is displayed in Table 5.3.

TABLE 5.3: Noise Addition for Sentences

Original Sentence	**Distorted Sentence**
Plaintiff argued that the district court decision was unreasonable.	Plaintiff **guilty** that the district court **an** was unreasonable.

50% of the sentence dataset is distorted by random replacement and the label 1 is assigned for each distorted sentence. Label 0 is assigned for each original sentence.

According to Fig. 5.10, The model is trained for sentence discrimination task and sentence pair similarity task at each training step. Model shares the same embedding layer and Recurrent Neural Network (RNN) layer for both tasks. Sentence Dataset provides a batch of sentences containing original and distorted sentences and from the Dense layer output, the probability of a sentence being either original or distorted is calculated. Discrimination loss is then calculated using this probability and the actual label. At the same training step, STS dataset provided a batch of sentence pairs and the similarity calculator produces the cosine similarity between the two vectors output by RNN layer. STS loss is calculated using the similarity value and the STS score

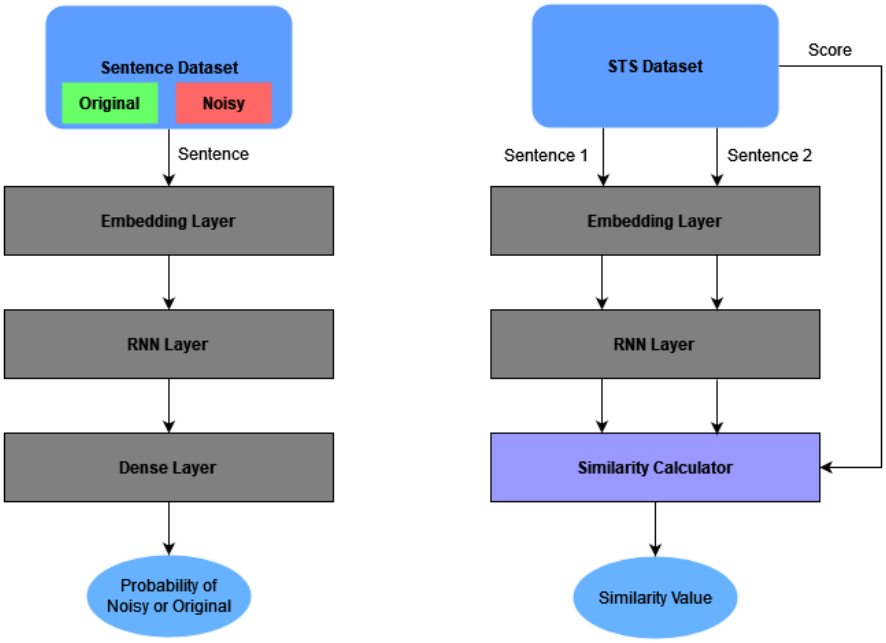

Fig. 5.10: Multi-task model Architecture

provided by the dataset. Model weights are optimized using both Discrimination loss and STS loss.

This multi-task approach aims to train the model to capture the semantics of the sentences while preventing the model from over-fitting for STS Dataset which is relatively small compared to Sentence Dataset. Discrimination task force the model to identify distortions only by looking at the sentence vector provided by the RNN layer. STS task trains the model to move vectors of similar sentences closer in the vector space and irrelevant sentences further away. This approach is suitable for a setting where a large corpus of unlabeled data is available along with a small set of labeled data and the annotation cost is high to expand the labeled dataset.

5.4 Winning Party Prediction

5.4.1 Dataset Preparation

The dataset used for this research component was extracted from the case law website [4] ranging from the years 2000 to 2010 in the criminal category. These extracted cases were then automatically pre-processed to remove the introductory paragraphs where the background of the case is stated and the last paragraphs where the decision is stated along with the footnotes and citations. During this process the decision of the court cases were extracted automatically from the final paragraph of the case. Afterwards

[4] https://caselaw.findlaw.com/

several preprocessing steps were applied to remove citations that may be incorrectly associated with the case data, as they do not give any semantic meaning to the case.

A case document in US Supreme Court generally consists of:

- Background information (represented Jury, Date of Hearing)
- A description of the scenario
    1. involved parties and their members (petitioners and defendants)
    2. how the case is formed (What caused the filing of the case)
    3. Available Evidence
    4. Lower court decision (Where the case was initially called)
- Supreme Court hearing
    1. Charges against the petitioner
    2. Opinions of jury
    3. Arguments bring forward by each party
- Footnotes

Cased documents were labeled by the decision with respect to the petitioner party. Affirmation, dismissal or rejection of a case by US Supreme Court results in petitioner losing. Reversal of a lower court decision results in petitioner winning the case.

5.4.2 Model Architecture

This section discusses the approach taken to predict the winning party of a court case using the available content of a case document. A case document is represented as a sequence of sentences and the model takes the corresponding sequence of sentence vectors as input. Additional information about a case sentence such as its critical aspect towards a party can be annotated using Critical Sentence Identifier which is described in Section 5.2. probabilities of the four classes provided by Critical Sentence Identification model are appended to sentence vector by increasing the dimension. Processed sentence vector sequence is then passed on to Document Encoder model which is configured using RNN or Transformer Encoder layers. Using the output of the Document Encoder model, probability of petitioner party winning or losing the case is obtained through a classifier component. This classifier component is configure using a Linear Neural Network. Overall workflow of the process is depicted in Fig. 5.11.

Considering the nature of a legal case, probability of Defendant party winning the case is equal to the probability of Petitioner party losing the case.

Internal Model architecture for RNN is displayed in Fig. 5.12 and for Transformer Encoder is displayed in Fig. 5.13.

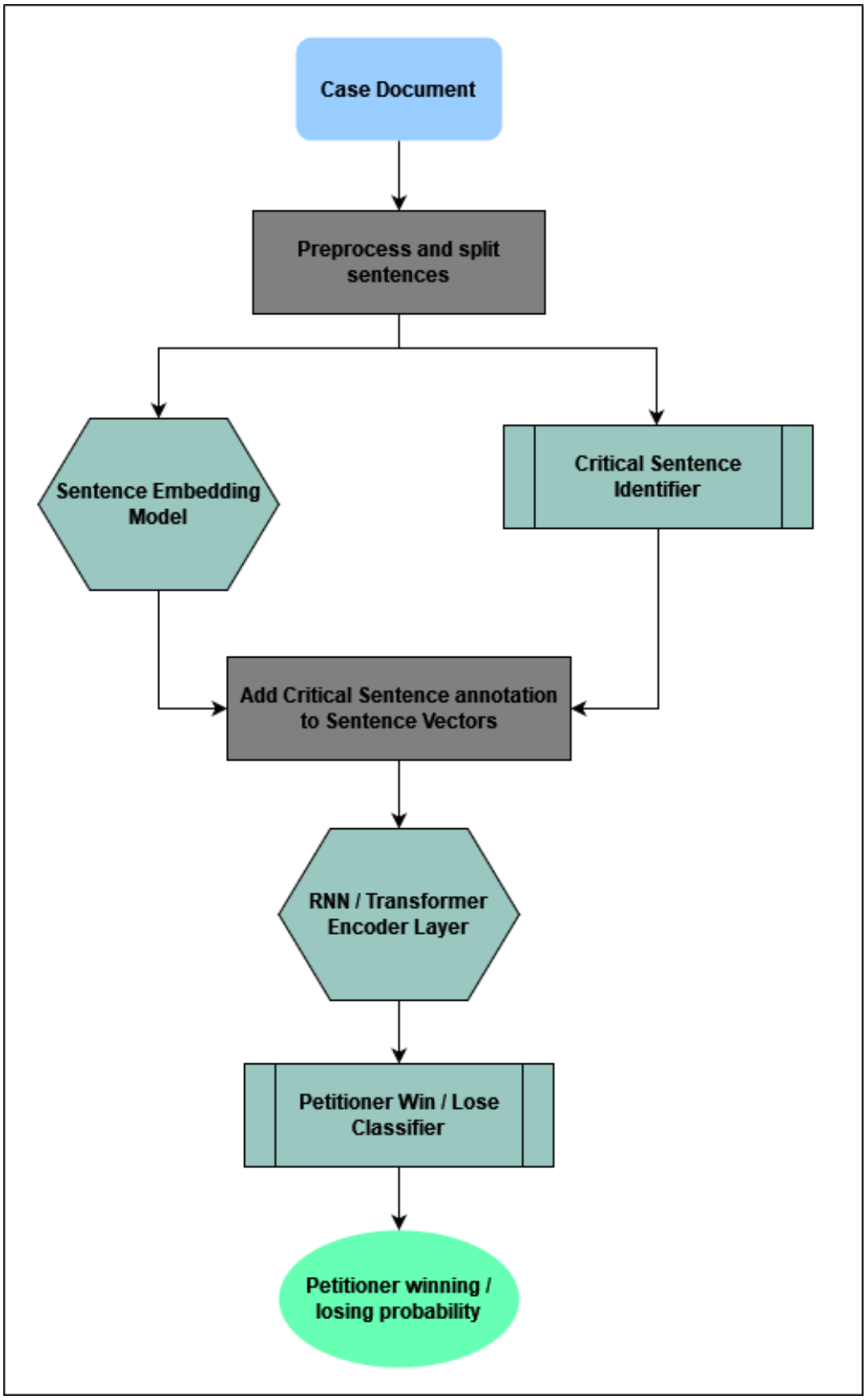

Fig. 5.11: Winning Party Prediction Workflow

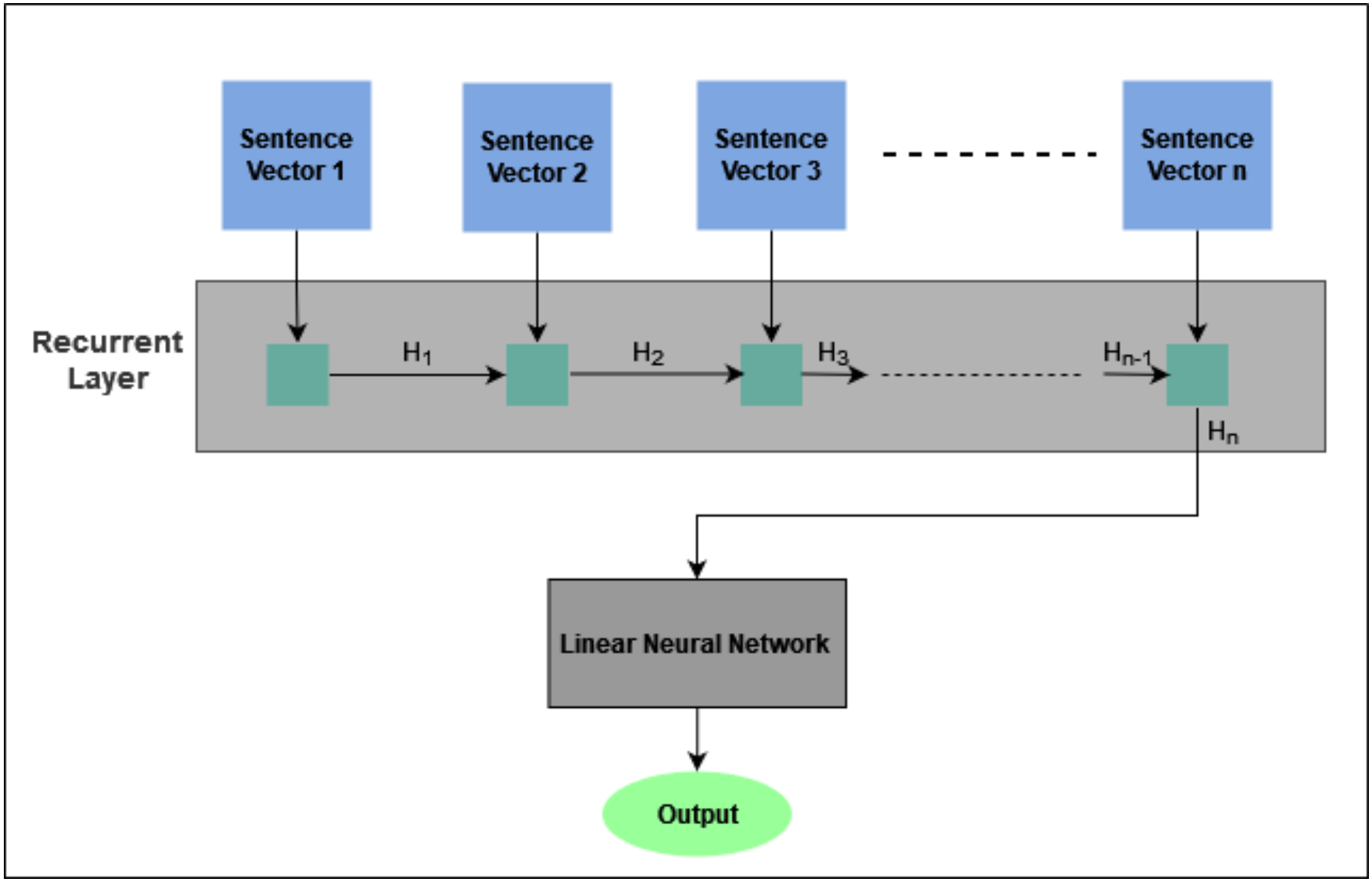

Fig. 5.12: Winning Party Prediction RNN Model

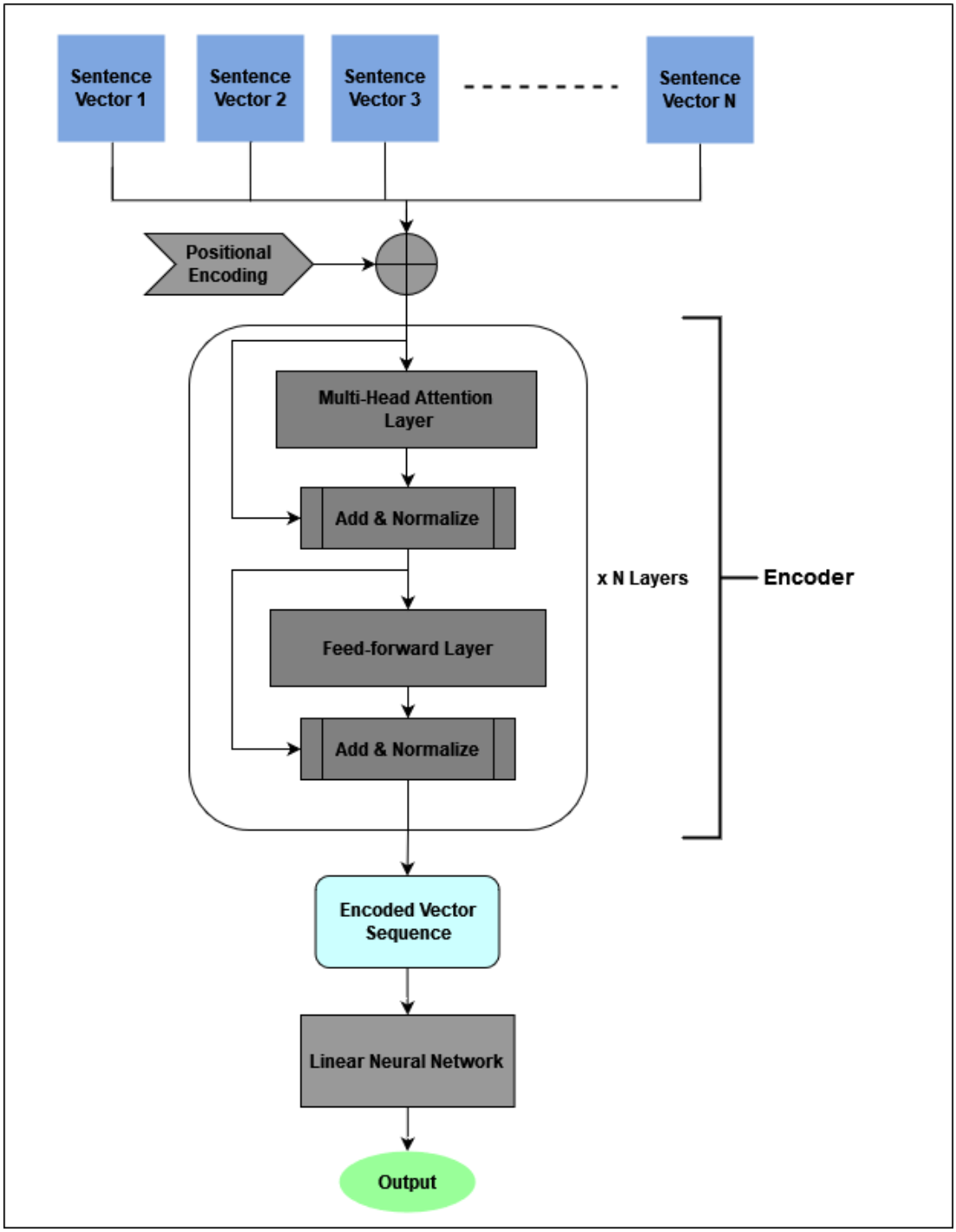

Fig. 5.13: Winning Party Prediction Transformer Encoder Model

CHAPTER 6

EXPERIMENTS AND RESULTS

6.1 Party-based Sentiment Analysis Pipeline for the Legal Domain

In order to ensure accurate input are fed into the *PBSA* implemented by Mudalige et al. [7], 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-RNN model which consists of GRUs with 512 output units, trained and evaluated as the best performing *PE* model [9] for the dataset of 1000 US supreme court case paragraphs. *Table 6.2* represents the metrics for the GRU 512 model configurations listed in *Table 6.1* for the last 100 paragraphs of the dataset.

TABLE 6.1: Models

Model	**Train**	**Evaluate**
Baseline - (Section 5.1.4)	original Co-Ref	original Co-Ref
nuRef - (Section 5.1.5)	original Co-Ref	updated Co-Ref
nuRefGRU - (Section 5.1.6)	updated Co-Ref	updated Co-Ref

TABLE 6.2: Metrics

Model	**Accuracy**	**Precision**	**Recall**	**F1**
Petitioner				
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%**
Defendant				
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%**

6.2 Critical Sentence Identification in Legal Cases

From the 1608 labeled case sentences we prepared as mentioned in Section 5.2.1, Train, Validation and Test set splits are created according to Table 6.3 . Over-sampling and under-sampling techniques are used to mitigate the class imbalance in the Train set.

In the over-sampling approach, samples from the 3 classes with lower sentence counts are duplicated to match the count of the *Petitioner win & negative* class of the train set. In contrast, the under-sampling method reduces the examples of 3 classes

TABLE 6.3: Dataset Statistics

Class	Original Train	Over-sampled Train	Under-sampled Train	Validation	Test
Petitioner lose & negative	176	547	176	25	25
Petitioner lose & positive	180	547	176	25	25
Petitioner win & negative	547	547	176	70	70
Petitioner win & positive	365	547	176	50	50

which consist of higher counts, to match with the class with lowest number of examples. These 2 combinations of the dataset are used to train the classification model, first with categorical cross entropy loss, then with the task-specific loss function. We have experimented the opposite class loss weight hyper-parameter and the metrics comparison is displayed in Table 6.4.

TABLE 6.4: Performance Metrics

Loss Function	**Opposite Class Loss weight**	Over-sampled		Under-sampled	
		Accuracy	**Macro-F1**	**Accuracy**	**Macro-F1**
Categorical Cross Entropy	N/A	68.82	67.86	58.24	59.99
Task-Specific Loss Function	1	72.94	70.39	63.53	64.68
	2	73.53	72.02	61.18	60.49
	3	71.76	70.97	62.94	63.68
	4	74.12	**73.62**	63.53	64.83
	5	74.12	73.26	**65.88**	**65.66**
	6	**75.29**	73.24	60.00	60.52
	7	74.71	73.57	59.41	60.90
	8	71.18	70.67	59.41	61.11

Each model is trained for maximum 8 epochs to avoid over-fitting. Afterwards the model weights with the best validation accuracy is used to evaluate on the test set. According to Table 6.4, *Task-Specific loss function* provided better optimization for the sentence classification when compared to the categorical cross entropy loss. *Opposite class loss weight* is configured as a hyper-parameter for training and the values 4, 5, 6 showed the best results for the test set. The increased accuracy of the task specific loss function can be intuitively explained as a result of considering domain dependent case decision polarity.

6.3 Domain Specific Sentence Embedding

In this section we discuss the experiments performed using different configurations for Auto Encoder model. Several variations were experimented with the choice of word embeddings trained on general corpus and legal domain corpus. Also few variations were also tested with the auto encoder model to identify relatively better combination.

6.3.1 Pre-trained Word Embeddings

For our experiments, 3 types of word embeddings trained on general corpora and the legal corpus of 10,000 US Supreme Court cases, are used to initialize the Embedding Layer of the Auto-Encoder model. Pre-trained word embeddings on general corpora are obtained from online sources.

- Word2Vec - trained on Wikipedia texts 2014 + Gigaword 5 [1]
- GloVe - trained on Google News Dataset [2]
- FastText - trained on Wikipedia 2017, UMBC web-based corpus and statmt.org news dataset [3]

Statistics of the case dataset used to train Word2vec, GloVe and FastText word embeddings as discussed in Section 5.3.1 are displayed in Figure 6.1.

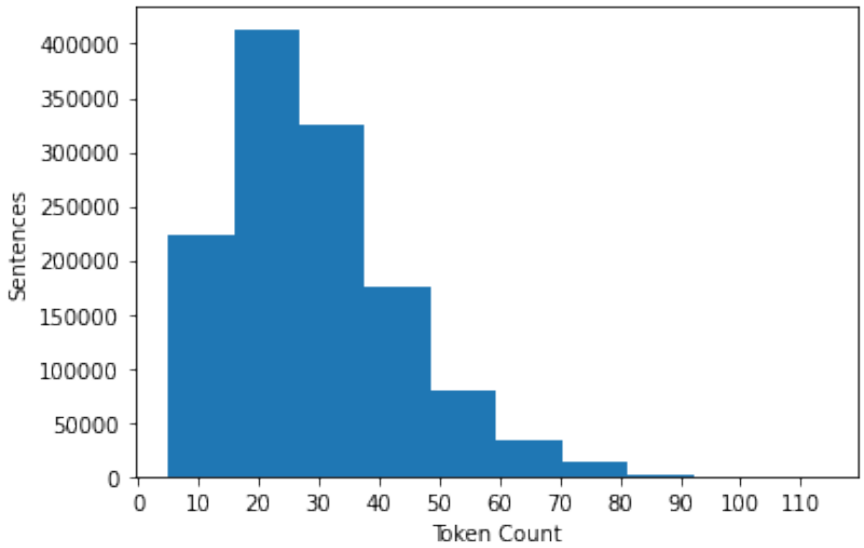

Fig. 6.1: Legal Case Dataset Statistics

[1] https://code.google.com/archive/p/word2vec/
[2] https://nlp.stanford.edu/projects/glove/
[3] https://fasttext.cc/docs/en/english-vectors.html

TABLE 6.5: Word Embeddings - Similar Words Comparison

Word	Glove	Word2Vec	FastText	Gl Legal	W2V Legal	FT Legal
Petitioner	respondent	appellant	respondent	habeas	defendant	defendant
	appellant	counsel	petition	plaintiff	appellant	appellant
	plaintiff	respondent	plaintiff	appellant	respondent	respondent
Lawyer	attorney	attorney	attorney	attorney	counsel	nonlawyer
	counsel	solicitor	ex-lawyer	client	attorney	attorney
	prosecutor	counsel	non-lawyer	appointed	client	cocounsel
Affirm	reaffirm	reaffirm	reaffirm	accordingly	reverse	reverse
	uphold	uphold	confirm	reverse	uphold	reaffirm
	reiterate	reiterate	assert	therefore	vacate	vacate

Figure 6.1 represents the number of tokens for 1262126 sentences extracted from 10,000 cases. 52% of the total sentences contain tokens within the range 20 - 40. We will be referring to the word embedding types trained on this legal corpus as *Word2Vec_Legal*, *GloVe_Legal* and *FastText_Legal* for the purpose of distinguishing them from word embeddings trained on general corpora.

Table 6.5 compares the most similar words predicted by the 6 types of Word Embeddings for 3 selected words in legal domain.

6.3.2 Auto-Encoder Results

1000 US Supreme court cases consisting of 125719 sentences are used for the training and evaluation of the Auto-Encoder model. Experiments are done based on the word embedding type and Recurrent layer type. All the variations listed in Table 6.6 are trained for 20 epochs and measured the results as a controlled experiment to choose the relatively best variation for further training.

6.3.3 Multi-task Model

Multi-task Model of Noise Discrimination and STS tasks is trained and evaluated using different configurations of GRU and LSTM layers. Accuracy and F1 scores are recorded for Noise Discrimination task and STS evaluation results are recorded using Pearson and Spearman Correlation between predicted similarity value and the STS score. Table 6.7 displays the results.

TABLE 6.6: Auto Encoder Metrics

RNN Type	Units	Word Embedding	Train Cosine Sim.	Validation Cosine Sim.
GRU	512	Glove	0.2663	0.2578
		Word2Vec	0.2068	0.2033
		FastText	0.3323	0.3299
		Glove Legal	0.2776	0.2743
		Word2Vec Legal	0.2358	0.2310
		FastText Legal	0.2182	0.2153
Bi-GRU	512	Glove Legal	0.2550	0.2499
		Word2Vec Legal	0.2249	0.2180
		FastText Legal	0.2139	0.2097

TABLE 6.7: Multi-task Model Metrics

RNN Type	Units	Accuracy	F1 (Original)	F1 (Noisy)	Pearson C.	Spearman C.
GRU	512	92.21	91.94	92.46	46.81	48.09
GRU	768	93.70	93.71	93.70	57.62	51.34
LSTM	512	89.68	89.36	89.98	39.93	35.25
LSTM	768	92.73	92.52	92.92	34.35	28.76

6.4 Winning Party Prediction

The number of layers for the transformer encoder was experimented with values 6,3,2,1. As seen in the figure 6.2 the best number of layers for the transformer encoder was identified to be one. RNN and Transformer Encoder components are used to encode case documents. RNN models are experimented using GRU and LSTM layers.

Transformer Encoder configuration used:

- Number of Encoder layers = 1
- Number of Attention Heads = 8
- Vector Dimension = 768

Binary Focal Loss [43] is used to calculate the loss at each train step due to its ability to handle class imbalanced datasets. At each training step, Focal Loss down-weights the loss for examples classified with higher accuracy of the dominant class and up-weights the loss for incorrectly classified examples of the minority class.

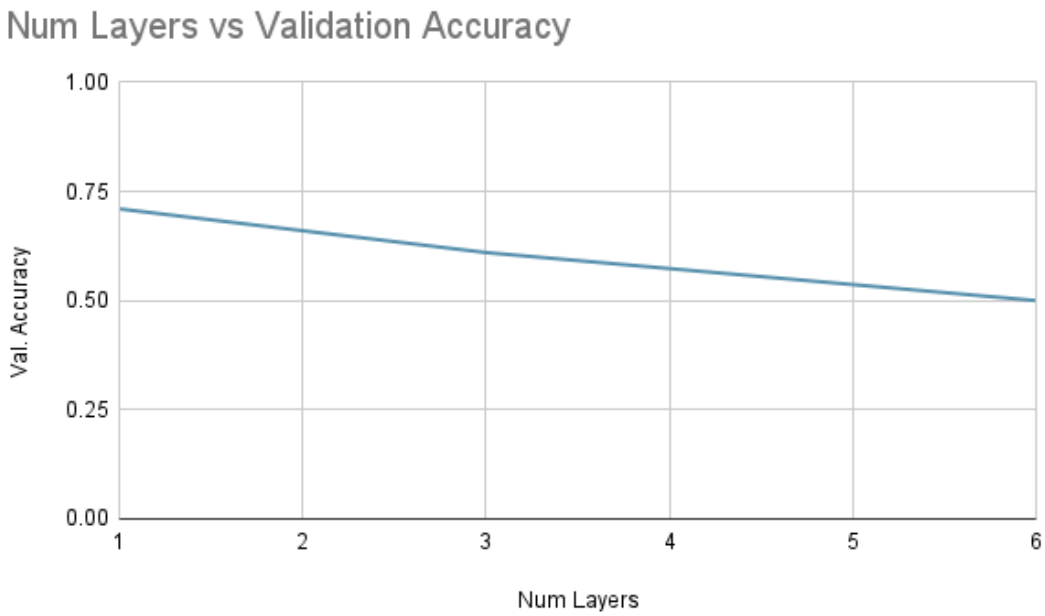

Fig. 6.2: Number of Layers vs Validation Accuracy

TABLE 6.8: Winning Party Prediction Metrics

Model	**Sentence Embedding**	**Critical Sentence Annotation**	**Accuracy**	**Macro F1**
GRU	bert-base-nli-mean	N	56.32	53.14
	distil-roberta-base	N	65.71	57.14
	distil-roberta-base	Y	73.05	63.27
LSTM	distil-roberta-base	Y	72.04	65.52
GRU - Bidirectional	distil-roberta-base	Y	75.46	63.88
Transformer Encoder	bert-base-nli-mean	N	69.26	60.85
	distil-roberta-base	N	74.88	64.96
	distil-roberta-base	Y	**75.75**	**66.54**

Table 6.8 elaborates the experiments performed to identify the best performing Winning Party Prediction model and their respective results. Different sentence embedding models are used along with critical sentence annotation. Marginally higher results are shown when critical sentence predictions are appended to sentence vectors.

CHAPTER 7

PUBLICATIONS

7.1 ICTer 2021 : "Party-based Sentiment Analysis Pipeline for the Legal Domain"

Conference : 21st International Conference on Advances in ICT for Emerging Regions (ICTer) 2021

Abstract : Since the advent of research to automate existing manual language workflows, legal domain has risen as a key area. When it comes to making the process of court cases more efficient with the said automation, legal information extraction is vital for legal professionals. In this study, we propose a unified pipeline to annotate party-based sentiment for court cases which reduces manual work. To achieve this end, we combined two state-of-the-art models into a single workflow. The first model extracts the entities which represent each party in a court case, while the second model analyzes the sentiment with respect to each party in a given sentence of a court case. In this study we propose two approaches for defining the pipeline which maps the output of the party extraction system into the party-based sentiment analysis system. Further, we propose improvements to the existing party extraction system.

State of the Paper : Published

7.2 ICIIS 2021 : "Critical Sentence Identification in Legal Cases Using Multi-Class Classification"

Conference : IEEE 16th International Conference on Industrial and Information Systems (ICIIS) 2021

Abstract : Inherently, the legal domain contains a vast amount of data in text format. Therefore it requires the application of Natural Language Processing (NLP) to cater to the analytically demanding needs of the domain. The advancement of NLP is spreading through various domains, such as the legal domain, in forms of practical applications and academic research. Identifying critical sentences, facts and arguments in a legal case is a tedious task for legal professionals. In this research we explore the usage of sentence embeddings for multi-class classification to identify critical sentences in a legal case, in the perspective of the main parties present in the case. In addition, a task-specific loss function is defined in order to improve the accuracy restricted by the straightforward use of categorical cross entropy loss.

State of the Paper : Published

CHAPTER 8

DISCUSSION

The primary objective of this research has been to design and develop a methodology to predict the winning party of a court case. Doing so it was identified that giving the visibility of the court case through multiple channels would be beneficial to the final winning party prediction model. Therefore we have developed such models with the intention of integrating them into the final model. The PBSA pipeline, Critical sentence identification model, Domain Specific sentence embedding model are such components. Also we have designed trained and evaluated such a model architecture, compatible with the extension with such models with sufficient accuracy.

The PBSA pipeline have reduced the manual work that would have been required otherwise when using the Party Identification and PBSA models separately. Also it has been improved to cater for several inefficiencies. The critical sentence identification model provides with the insight to the final model on the criticality of a sentence towards the overall case decision. This has hopefully given the final model more visibility on the court case in a sentence wise manner. Also, domain specific sentence embedding methods were tried out with different approaches to address the domain specific complexities in contrast to the general purpose sentence embeddings. They can be further improved and integrated to the final model for better accuracy. The final winning party prediction model supports the integration of the aforementioned models and other models that may be developed in the future. They can be added as an additional dimension to the input vectors before passing in to the final model.

The 1st phase of our project is about developing an automated pipeline for party based sentiment analysis which mitigates the manual annotation of party members when using PBSA system independently. Pipeline is built by integrating the Party Identification subsystem developed by Samarawickrama et al. [10] and Party-based Sentiment Analysis subsystem developed by Rajapaksha et al. [8] into a unified workflow. Further improvements are made to Party Identification subsystem to mitigate an issue occurred by Stanford Co-reference annotation which led to incorrect input preparation for PBSA subsystem. This improvements altogether improved accuracy of the process as well. Two approached for designing the pipeline is implemented and algorithms and diagrams are elaborated. PBSA pipeline outputs the sentiments for each mentioned member in petitioner and defendant parties for a given case sentence which provides the user, insights about the impacts of the arguments brought forward in a court case.

In the 2nd phase of the project, we developed a model to predict the critical nature of a case sentence which is based on the likely decision outcome and the sentiment towards the petitioner party as a whole. A case sentence is considered critical when

the sentiment is positive towards petitioner in a case where petitioner wins and when the sentiment is negative towards petitioner in a case where petitioner loses. By annotating the decisions and overall sentiments towards petitioner, we extended the PBSA dataset [7] and trained a multi-class classification model. Furthermore, a loss function is defined specifically for this task which eventually outperformed the categorical cross entropy loss in this specific use case. The critical sentence identification model is integrated into winning party prediction system where it's used for providing insightful information on each sentence of a case document.

Approaches for training a sentence embedding model for legal domain datasets are explored in the 3rd phase of our project. Through web scraping, we extracted a set of 10,000 US Supreme court case documents and then we focused on unsupervised techniques for training sentence embeddings using the large corpus of unlabeled data. First we experimented an Auto Encoder model where the Encoder is responsible for providing a unified vector for the input sentence and the decoder reconstructs the original sentence token by token. Then we developed a STS labeled dataset along with the help of a professional in legal domain. Using this labeled dataset and the large unlabeled corpus, we designed a multi-task model for training sentence embeddings. The two tasks defined are to capture the semantic meaning using the STS label between a pair of sentences and a noise discrimination task to identify whether a sentence is distorted or not. Our approach addresses a situation where a large set of unlabeled data is available along with a small set of labeled data to train a sentence embedding model.

In the final phase, we developed a workflow for predicting the winning party of a court case. This system incorporates the critical sentence identifier model to provide additional information regarding each sentence of the case document. Document is represented as a sequence of sentence vectors and a document encoder is trained followed by a classifier neural network to predict the probability of petitioner winning the case. Experiments are performed using different document encoder architectures and sentence embedding models. Inclusion of critical sentence model to the workflow proved valuable according to the experimental results.

CHAPTER 9

FUTURE WORK

Since the final winning party prediction model accuracy is reliant on the accuracy of the sub-components it is important to conduct more research to improve the accuracy of them. Also the final model is extensible by other models that may provide visibility to the winning party prediction model. Therefore it is important to conduct research to develop such models. Such components can be designed to either derive intuitive information about the case or extract valuable information from the case, so that it contributes to making the prediction more accurate. Examples of such information are evidence vectors, evidence counts, related and cited case content vectors. It should be kept in mind to design these sub components to provide perspective over facts that may contribute to the final case decision.

The sentence embedding model can be considered the most important sub-component as it contributes the most dimensions to the input vector of the final system. Therefore it should be improved for better accuracy in the legal domain. Due to the availability of a large legal corpora and the limitations of labeled data in the legal domain, it is important to conduct further research on domain specific sentence embeddings in unsupervised approaches.

Furthermore it would be beneficial to legal professionals if they could trace back to the case content that contributes most to the case decision based on empirical data; that is the case law documents. If the model can accurately back trace the decision to the important facts or arguments of the case it would help legal professionals prepare for similar cases with similar arguments and for academic purposes of legal students as well. Therefore, that is another future direction that research on winning party prediction of legal cases can be extended to.

CHAPTER 10

INDIVIDUAL CONTRIBUTIONS

TABLE 10.1: Individual Contributions

Phase	**Tasks Carried out**	**170264**	**170481**	**170584**
Project Proposal	Writing the Introduction	-	100	-
	Writing the problem statement	100	-	-
	Formulating the Research Objectives(ROs)	40	30	30
	Reading related work and writing the Literature Review	30	40	30
	Formulating and writing the methodology	30	30	40
	Feasibility, Timeline and conclusion	-	-	100
RO1 : Party Based Sentiment Analysis pipeline (ICTer2021)	Implementing the Baseline model with basic PBSA and Party Identification Models	80	10	10
	Implemented the improvements to the party identification process	50	10	40
	Retrained the party identification model with improvements	20	20	60
	Implemented the Adapter to Connect the two subsystems	60	40	-
	Experimenting and evaluating improvements to the Party Identification model	30	20	50
	Writing the paper for ICTer 2021	40	50	10
RO2 : Critical Sentence Identification (ICIIS2021)	Preparing the dataset with sentiment average and case decision annotation using the PBSA dataset	60	-	40

	Exploratory Data analysis with visualizations	40	50	10
	Designing the model architecture	80	-	20
	Implementing the task specific loss function	100	-	-
	Conducting experiments on the model and evaluating the loss function against Cross entropy loss.	30	20	50
	Writing the paper for ICIIS 2021	40	50	10
RO3 : Domain Specific Sentence Embedding	Web scraping US supreme court cases	-	100	-
	Preprocessing court cases by analyzing the case structure and domain specific requirements	33	33	33
	Automating the Court Case decision annotation.	-	100	-
	Train legal domain word embeddings with Court case dataset for Glove, Word2Vec and FastText	33	33	33
	Designing the Auto Encoder architecture for sentence embeddings	100	-	-
	Experimenting the Auto Encoder using general word embeddings and legal domain word embeddings.	40	30	30
	Preparing NLI and STS dataset	20	80	-
	Designing multi-task model architecture for learning sentence embeddings.	50	-	50
	Conducting experiments using STS labeled dataset and unlabeled case sentence dataset.	40	-	60

	Writing the research paper on a self-supervised approach for sentence embedding.	40	50	10
RO4 : Winning Party Prediction	Preprocess court case documents by removing starting sections of background info and footnotes.	30	40	30
	Representation of a court case document - saving sentence embedding sequence	-	-	100
	Executing critical sentence identification on court case sentences and saving the outputs.	-	50	50
	Designing the architecture of the winning party prediction model.	100	-	-
	Implementing RNN based model for winning party prediction	-	-	100
	Implementing Transformer Encoder based model for winning party prediction.	50	50	-
	Training the models for different configurations and evaluating.	33	33	33
	Experiment and evaluating the effect of using critical sentence prediction data as additional dimensions for sentence vectors.	33	33	33

REFERENCES

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

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] C. Samarawickrama, M. de Almeida, A. S. Perera, N. de Silva, and G. Ratnayaka, "Identifying legal party members from legal opinion texts using natural language processing," EasyChair, Tech. Rep., 2021.

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

[12] 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.

[13] ——, "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.

[14] 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.

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

[16] 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.

[17] 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.

[18] R. A. Shaikha, T. P. Sahua, and V. Anand, "Predicting Outcomes of Legal Cases based on Legal Factors using Classifiers," *Procedia Computer Science 167 (2020) 2393–2402*, 2020.

[19] N. Aletras, D. Tsarapatsanis, D. Preoţiuc-Pietro, and V. Lampos, "Predicting judicial decisions of the european court of human rights: a natural language processing perspective," *PeerJ Computer Science*, vol. 2, p. e93, Oct. 2016.

[20] Y. Liu and Y.-L. Chen, "A two-phase sentiment analysis approach for judgement prediction," *Journal of Information Science*, vol. 44, 07 2017.

[21] D. M. Katz, M. J. Bommarito II, and J. Blackman, "Predicting the behavior of the supreme court of the united states: A general approach," *arXiv preprint arXiv:1407.6333*, 2014.

[22] A. Lage-Freitas, H. Allende-Cid, O. Santana, and L. de Oliveira-Lage, "Predicting brazilian court decisions," *arXiv preprint arXiv:1905.10348*, 2019.

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, 2013, pp. 3111–3119.

[24] 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.

[25] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[26] I. Glaser, E. Scepankova, and F. Matthes, "Classifying semantic types of legal sentences: Portability of machine learning models," in *Legal Knowledge and Information Systems*. IOS Press, 2018, pp. 61–70.

[27] J. Jagadeesh, P. Pingali, and V. Varma, "Sentence extraction based single document summarization," *International Institute of Information Technology, Hyderabad, India*, vol. 5, 2005.

[28] T. Hirao, H. Isozaki, E. Maeda, and Y. Matsumoto, "Extracting important sentences with support vector machines," in *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[29] 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)*, 2021, pp. 171–176.

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

[31] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based lstm for aspect-level sentiment classification," in *EMNLP*, 2016, pp. 606–615.

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

[33] S. Jayasinghe, L. Rambukkanage, A. Silva, N. de Silva, and A. S. Perera, "Critical sentence identification in legal cases using multi-class classification," in *2021 IEEE 16th International Conference on Industrial and Information Systems (ICIIS)*. IEEE, 2021, pp. 146–151.

[34] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[35] D. Datta, P. E. David, D. Mittal, and A. Jain, "Neural machine translation using recurrent neural network," *International Journal of Engineering and Advanced Technology*, vol. 9, no. 4, pp. 1395–1400, 2020.

[36] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[37] 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.

[38] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[39] A. Conneau and D. Kiela, "Senteval: An evaluation toolkit for universal sentence representations," *arXiv preprint arXiv:1803.05449*, 2018.

[40] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation," *arXiv preprint arXiv:1708.00055*, 2017.

[41] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[42] J. Huang, D. Tang, W. Zhong, S. Lu, L. Shou, M. Gong, D. Jiang, and N. Duan, "Whiteningbert: An easy unsupervised sentence embedding approach," *arXiv preprint arXiv:2104.01767*, 2021.

[43] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.