Hybrid CNN: An Empirical Analysis of Machine Learning Models for Predicting Legal Judgments

G. Sukanya[1], J. Priyadarshini[2]*
School of Computer Science and Engineering, Vellore Institute of Technology,
Chennai Campus, Chennai, India[1, 2]

***Abstract*—Artificial Intelligence with NLP has revolutionized the legal industry, which was previously under-digitized, and it's eager to adopt digital technologies for increased efficiency. Case backlog issues, exacerbated by population growth, can be alleviated by AI's potential in decision prediction for laypeople, litigants, and adjudicators. Legal judgment prediction (LJP) is viewed as a text classification cum prediction problem, with encoding models crucial for accurate textual representation and downstream tasks. These models capture syntax, semantics, and context, varying in performance based on the task and dataset. Selecting the right model, whether traditional ML or DL, using different evaluation metrics, is complex. This paper addresses the above research gap by reviewing 12 cutting-edge ML models and 10 DL models with two embedding methods on real-time Madras High Court criminal cases from Manupatra. The comprehensive comparison of classifier models on real-time case documents provides insights for researchers to innovate despite challenges and limitations. Evaluation metrics like accuracy, F1 score, precision, and recall show that Support Vector Machines (SVM), Logistic Regression, and SGD with Doc2Vec (D2V) encoding and shallow neural networks perform well. Although Transformers process longer input sequences with parallel word analysis and self-attention layers, they have weaknesses on real-time datasets. This article proposes a novel hybrid CNN with a transformer model to predict binary judgments, outperforming traditional ML and DL models in precision, recall, and accuracy. Finally, we summarise the most important ramifications, potential research avenues, and difficulties facing the legal research field.**

Keywords*—Legal judgment prediction; encoding; SVM; SGD; Doc2vec; CNN; transformers

I. Introduction

Legal Artificial Intelligence is a fast expanding field that includes managing and analyzing massive amounts of legal documents with the aid of cutting-edge algorithms and machine learning techniques. Text classification is a versatile and powerful technique that can automate and streamline various aspects of information management, decision-making, and user interaction in a wide range of industries and applications. It enables organizations to extract valuable insights from text data and enhance their operational efficiency and user experience. Processing casefacts based on final judgment has always been done manually. However, it can be highly expensive and takes up a lot of the time of the personnel, if done manually. Automated text classification technologies can be of great assistance in this situation. To efficiently structure and analyse massive amounts of text, we integrate NLP and machine learning models. Though the work involves preprocessing steps of raw legal documents it emphasizes on the effect of different Word embedding on classifier models. As maximum information loss occurs at encoding stage and, only a few studies have focused on identifying the influence of the features and interpreting the machine learning models, the comprehensive comparison done in this survey would be an eye-opener for researchers in the field of Natural Language Processing. Legal Judgment Prediction is generally considered as a text classification cum prediction. Fig. 1 sketches the general steps in machine learning and deep learning models used in the prediction of judgment whether the case is allowed or dismissed based on the preprocessed casefacts. In both machine learning and deep learning models, the initial raw case document undergoes preprocessing steps such as word-level tokenization, lemmatization and stemming. Feature extraction using count-based models is done using lemmatized case document. In the deep learning model, feature extraction is done with dense embedding and hidden layer.

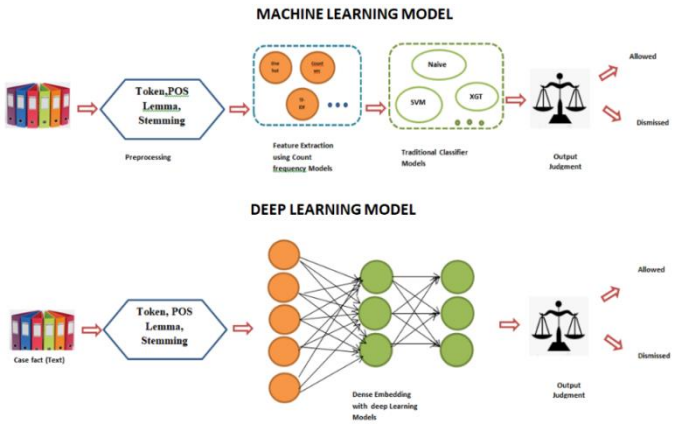

Fig. 1. Text classification using ML and DL model.

A. Benefits of ML and DL Models in LJP

- Improved Accuracy and Consistency
- Efficiency and timesaving
- Cost-effectiveness
- Data driven Insights are better

B. Limitations of Legal Judgment Prediction

The limitations on the study of legal judgment prediction are listed in the following points.

- Ambiguity in Legal Language

- Context Sensitivity
- Lack of Standardization
- Imbalanced Datasets.
- Data Annotation Challenges
- Lengthy Documents
- Lack of experimental analysis with real time casefacts using Machine Learning and Deep Learning models

As most of the existing research works are on legal judgment prediction with synthesized Chinese court case datasets and they do not emphasize on different encoding methods which results in information loss, our study with proposed Hybrid CNN model addresses the effect of different embedding methods thereby addressing the above research gap.

The aim of this systematic review is to determine the best ML and DL model and compare its performance based on different embedding methods and it also highlights the impact of using Transformer along with CNN for text documents. Specifically, this review aims to answer the following research questions given in Table I.

TABLE I. RESEARCH QUESTION

ID	Question
RQ 1	Which Machine Learning and Deep Learning models perform better to classify real time case documents for judgment classification
RQ 2	Does change of encoding models have an impact on prediction accuracy?
RQ 3	Comparison of the baseline ML and DL methods and encoding methods based on evaluation metrics
RQ 4	Importance of proposed hybrid CNN with transformer model

The following section of this article is organized as follows. : In Section II, we discuss related works on legal Judgment prediction. Section III outlines the experimental ML and DL methods. Section IV outlines the hybrid CNN model with a transformer. Section V presents the performance analysis and Section VI concludes the paper and suggests directions for future works.

II. RELATED WORKS

Predictive analytics and NLP have seen significant exploration in the legal domain. Kort [1] underscored the role of quantitative analysis in predicting US Supreme Court outcomes. Studies like those by Octavia-Maria et al. [2] and Katz et al. [5] have further validated machine learning for legal predictions, using SVM classifiers to forecast French Supreme Court decisions and a time-evolving random forest classifier to predict US Supreme Court behavior, showcasing machine learning's utility in legal analysis.

Legal prediction models now cover a wider range of scenarios, thanks to recent developments in deep learning. Recent advancements in deep learning have expanded the scope of legal prediction models. Chalkidis et al. [3] explored neural models for judgment prediction, demonstrating their superiority over traditional methods. Kaufman et al. [6] introduced AdaBoosted Decision Trees for forecasting US Supreme Court decisions, achieving remarkable accuracy by incorporating textual data from oral debates.

Furthermore, the advent of pre-trained language models like Devlin et al., [7] has introduced BERT which revolutionized natural language understanding tasks. Chalkidis et al. [8] developed LEGAL-BERT, a domain-specific language model trained on legal texts, highlighting its potential to enhance legal text analysis accuracy. Attention mechanisms have also been utilized for legal judgment prediction, as demonstrated by G. Sukanya and Priyadarshini J. [15], proposed a Modified Hierarchical Attention Network (MHAN) for precise outcome prediction using hybrid classifier in Indian judicial cases.

While these studies showcase the immense potential of machine learning and NLP in legal analytics, challenges such as model interpretability, bias in training data, and adaptability across legal systems and languages remain significant concerns [4] [13]. Also, the application of real-time court cases on all traditional machine learning with different encoding methods and deep learning models is yet to be explored. Furthermore, the complexity of legal texts and the need for large, specialized datasets pose additional hurdles in model development and evaluation [9] [10] [11] [12]. Also, most of the existing works were done using Chinese cases such as the CAIL dataset [20] [21] [22] only and very few using other case datasets [23] such as ECHR (European Convention of Human Rights).

Vaswani et al. [16] introduced the importance of attention mechanism in Transformer, a novel neural network architecture that relies solely on attention mechanisms, eliminating the need for recurrent or convolutional layers. This design allows for increased parallelization, significantly reducing training time. The Transformer model achieves state-of-the-art performance on machine translation tasks, demonstrating its effectiveness and efficiency. Furthermore, it generalizes well to other tasks like English constituency parsing, showcasing its versatility and potential for a wide range of applications.

Existing works which explained the transformer model briefly such as GPT-3, BERT [7], and T5 [17] provide an overview of Transformer models, highlighting their significance in machine learning, especially in NLP. It introduces transformers as a breakthrough architecture that outperforms earlier models, such as RNNs, by allowing data to be processed in parallel, increasing efficiency and model performance [16]. To help the Transformer comprehend context and relationships within data, the paper goes into detail on important ideas like positional encodings, attention mechanisms, and self-attention. The impact of transformers in a variety of applications—from content creation to language translation—as well as their part in the creation of cutting-edge models like GPT-3 and BERT are also covered.

To sum up, the amount of research on NLP and predictive analytics in legal settings shows how machine learning approaches are becoming more and more popular for use in courtroom decision-making. The ability to predict court

decisions with high accuracy has advanced significantly, with researchers having progressed from simple quantitative analyses to complex deep learning models. Nonetheless, there is still ongoing research being done on issues like domain adaptation, bias mitigation, and model interpretability. Future research should concentrate on creating more reliable and interpretable models that can handle the complexity of legal texts and adjust to different legal systems and languages as the field develops.

A. Word Embeddings for Judgment Prediction

The core of any NLP assignment is word embeddings since they let the computer comprehend human language. The semantic content of text can be captured via word embedding, which is essential for text representation, as there is problem of polysemy words. Word embedding technique has been applied to the text classification and prediction task in numerous research [14]. Word embeddings map the words or phrases from vocabulary into vectors or real numbers. They mainly help in feature extraction for text related tasks.

Count based models such as one hot encoding, TF-IDF are generally used in traditional machine learning models. They work mainly on frequency of the words and do not deal with semantic representation of the word [18]. They are high dimensional and does not fulfill maximum representation of the document. Nowadays word embeddings such as Word2vec, Glove, Elmo, Doc2vec with shallow deep learning model to represent word into real valued vectors are used widely. They are again divided into static and dynamic word embeddings. The vector representation of any word is constant and does not change depending on the context for static word embedding models. In dynamic ones, a word's vector representation is generated using the context in which it is currently situated and changes as the context does. They represent the word by considering syntactic and semantic criteria. Choosing a correct word embedding for our application is also a tedious task, and this can be done by experimenting it with different word embedding in different Machine learning models. Sometimes customized word embeddings are used for certain specific applications as word embedding generally uses English words used in websites and ebooks which results in more Out of Vocabulary words during embedding.

III. Dataset Description

The dataset is prepared from 1466 unique real time raw case documents which comprises 15 types of criminal cases judged by Madras High Court for making legal predictions using Data Analytics and Machine Learning. The dataset is web scraped from the Manupatra website. Pipeline architecture has been used which uses text preprocessing stages such as removal of stopword, tokenization, stemming and lemmatization to convert the web scraped document into .csv file. With 36 distinct features, this dataset offers a comprehensive range of information, enabling in-depth analysis and insights into legal matters. Researchers, legal professionals, and analysts can leverage this dataset to explore patterns, trends, and correlations within the legal domain, contributing to advancements in legal research, policy formulation, and decision-making processes. Named Entity Recognition from SpaCy ,Genism Doc2Vec and Logistic Regression from sklearn models were used for Information Extraction. Table II shows the details of types of files.

TABLE II. Types of Files Extracted

File Type	Purpose
JSON	Legal Entities particularly Sections, Acts and Articles generated using Named Entity Recognition (NER) and Case Metadata obtained using String Matching Techniques and Regular Expressions.
TXT	All text data from the legal documents including facts, judgments and casenotes at different stages of preprocessing such as removal of stopwords, tokenization, stemming and lemmatiation are denoted as N1, N2, N3and N4.
CSV	All single valued columns comprising of numerical data, file paths and other case data

The relative file path of the lemmatized case facts, sections, acts, articles and judgments are the features used in experimental analysis of classifier models. Since the data available is not annotated, it was complex to extract more number of features from the dataset.

IV. Experimental Analysis

Experimental analysis of machine learning and deep learning models is an indispensable part of the data science and AI landscape. It involves the systematic investigation of these models to understand their performance, strengths, and weaknesses especially for using real-time dataset. This empirical approach is crucial in fine-tuning model parameters, ensuring robustness and selecting the most appropriate model for a given problem, ultimately advancing the field of artificial intelligence and machine learning.

In this section, the experimental setup using real time Madras High Court dataset with existing machine learning models and deep learning models is explained briefly and the outcomes are assessed using the evaluation metrics such as precision, recall and F1 score. Various encoding techniques, including Count Vectorizer, TF-IDF, and Doc2vec, were applied to 12 machine learning models which include SVM, KNN, MNB, Gradient Boost, Catboost, Adaboost, Random Forest, and Extra Trees. Additionally, deep learning models, including Shallow Neural Network, Deep Neural Network with varying numbers of dense layers, CNN, LSTM,GRU, Bidirectional GRU, Bidirectional LSTM, and Recurrent CNN with doc2vec embedding, were also examined.

Following Fig. 2 and Fig. 3 depict the detailed process flow of experimental setup of LJP using classifiers.

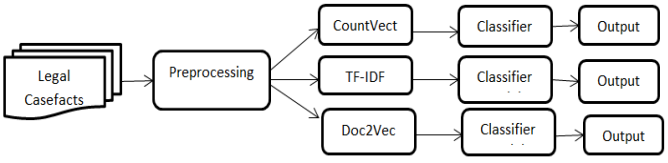

Fig. 2. Workflow process using machine learning models.

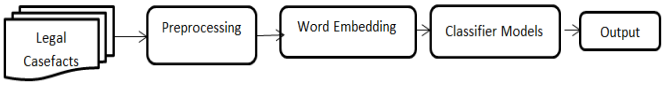

Fig. 3. Workflow process using deep learning models.

Raw case documents are taken as input, which is then made to undergo preprocessing steps such as removal of stopwords, tokenization, stemming and lemmatization. The lemmatized casefacts without judgment is considered as input and the tokenids are used for training the classifier model. Machine learning models are made to undergo different frequency based encoding methods such as CountVectorizer,TF-IDF(Term Frequency-Inverse Document Frequency) and Doc2Vec after tokenization, whereas deep learning models used word2vec embedding vectors for each token. Models are fed tokenized text together with matching labels or objectives for classification tasks during the training phase. The models discover relationships and patterns between the tokens and the intended result. Each of the Machine learning model and deep learning model used in the empirical analysis is explained in the paragraphs below. Nowadays transformers are used widely to void long range dependencies and for contextual representation. The use of subword tokenization concept in transformers manages to give some real vector value even for Out Of Vocabulary words. Finally a hybrid model with 1D CNN for feature extraction and transformer encoding to train the model is used.

A. Machine Learning Models

1) Support Vector Machine (SVM): SVM is a supervised model used for text classification, sentiment analysis, and document categorization, effective in high-dimensional spaces like text data. It finds the optimal hyper plane in a high-dimensional space to separate different classes but can be computationally intensive for large datasets and less effective when features outnumber samples. Hybrid models augment SVM by using dense vectors for documents, enhancing semantic analysis and classification performance. Incorporating Term Frequency-Inverse Document Frequency (TF-IDF) emphasizes term relevance, improving feature selection and classification outcomes.

2) Logistic regression (Logistic): Logistic Regression, a statistical model, applies a logistic function to model binary outcomes in text classification, estimating the probability of document categorization. It's pivotal for binary tasks like spam detection and sentiment analysis but assumes linear relationships between variables and outcomes, a limitation in complex text scenarios. Hybrid models like Logistic D2V, CV, and TF-IDF enhance Logistic Regression by transforming text into features, leveraging unique advantages of each method to boost classification accuracy by capturing detailed textual information and prediction models for judgment.

3) Gradient Boosting (GB): Gradient Boosting, an ensemble method, iteratively corrects errors of preceding models using decision trees as base learners. Widely used, it enhances accuracy in tasks like text classification by leveraging structured feature representations. However, it demands significant computation and is prone to overfitting without careful parameter tuning. Hybrid models like GB D2V and GB TF-IDF merge Gradient Boosting corrective approach with text features, enhancing performance on text-centric datasets.

4) CatBoost: CatBoost, a gradient boosting method designed for speed and accuracy, excels at managing categorical variables and reducing overfitting, making it ideal for complicated datasets such as text. While the inherent capability for categorical data improves efficiency, precise parameter adjustment is required. The hybrid models Catboost D2V, Catboost CV, and Catboost TF-IDF combine benefits of Catboost method with various text representation approaches to improve classification accuracy, especially in scenarios involving categorical data and text.

5) LightGBM (LGBM): LightGBM (LGBM) is a gradient-boosting framework renowned for its speed, economy, and accuracy in tree-based learning algorithms. It excels at processing massive amounts of data, including text classification jobs, and strikes a compromise between training speed and accuracy. However, it may overfit on smaller datasets and necessitate parameter adjustment. Hybrid models, such as LGBM D2V, CV, and TF-IDF versions, combine LGBM's efficient learning algorithm with various text representation approaches to improve performance in text classification tasks.

6) XGBoost (XGB): XGBoost, a distributed gradient boosting toolkit, is notable for its efficiency, versatility, and application to a variety of machine learning problems. It excels in handling sparse text data in classification applications, resulting in higher prediction accuracy. However, like LGBM, it is prone to overfitting and requires careful parameter tweaking. XGB D2V is a hybrid model that combines XGBoost with Doc2Vec's dense representations to create a powerful text classifier. By integrating XGBoost's efficiency with Doc2Vec's rich feature representations, predictive performance is improved while computational efficiency is maintained.

7) Extra trees: Extra Trees, or Extremely Randomized Trees, are similar to Random Forest but introduce more randomness in split and feature selection to reduce model variance. While effective for classification and regression tasks, particularly with text data, its random nature may occasionally result in lower accuracy compared to more refined ensemble methods. The hybrid models Extra Trees D2V, Extra Trees CV, and Extra Trees TF-IDF use various text preprocessing techniques to leverage the model's resistance to overfitting while handling the nuances of natural language.

8) Random forest: Random forests, an ensemble learning approach, train many decision trees and combine their results to produce predictions. They are effective for classification and regression applications, such as text classification, since they can handle high-dimensional data while minimizing overfitting. However, forecasting with huge numbers or deep trees can be sluggish, and sophisticated models can be difficult to understand. Hybrid models such as Random Forest D2V,

CV, and TF-IDF versions seek to enhance text data processing by using a variety of feature extraction approaches that capture both semantic and syntactic information.

9) Stochastic Gradient Descent (SGD): Stochastic Gradient Descent (SGD) is an optimization approach that iteratively adjusts model weights to reduce a loss function. It is especially useful for sparse machine learning applications such as text categorization. It excels at training linear classifiers like linear SVM and logistic regression on big datasets, but it necessitates precise hyperparameter and learning rate optimization. Sensitivity to feature scaling is an important concern. Hybrid models, such as SGD D2V, SGD CV, and SGD TF-IDF, use SGD to optimize linear models with different textual feature representations, striking a compromise between computational efficiency and classification accuracy.

10)AdaBoost: Adaboost, a boosting method, combines weak learners into more robust ones by modifying the weights of misclassified examples. It helps in text categorization by refining decision limits based on text complexity. However, it is susceptible to noise and may struggle if weak learners are extremely sophisticated. Hybrid models such as Adaboost D2V, Adaboost CV, and Adaboost TF-IDF improve text analysis by emphasizing difficult occurrences and improving categorization in complicated and high-dimensional text data.

11)K-Nearest Neighbors (KNN): KNN, a non-parametric method, categorizes documents by their nearest neighbours in feature space, useful for text classification where document similarity guides categorization. Yet, it's computationally demanding and sensitive to distance metric and K value. Hybrid Models like KNN D2V, KNN TF-IDF, and KNN CV leverage KNN on text data represented via Doc2Vec, TF-IDF, and Countvectorizer, respectively, offering varied ways to measure document similarity, potentially enhancing KNN's efficacy in classifying texts based on content likeness.

12)Multinomial Naive Bayes (MNB): A variant of Naive Bayes designed for multinomial distributed data. In text classification, it works well with the bag-of-words model where the frequency of words is used as feature. Particularly effective for document classification and spam filtering, where the independence assumption of features (words) holds reasonably well. The assumption of independence among features is often violated in text data, which can limit its effectiveness in more complex classification tasks. By combining MNB with TF-IDF and Count Vectorizer, these hybrids aim to enhance the model's ability to prioritize relevant features in text data, potentially improving classification outcomes.

B. Deep Learning Models

1) Gated Recurrent Unit (GRU): GRU, or Gated Recurrent Unit, is a form of recurrent neural network (RNN) that addresses the vanishing gradient problem which suffers from long-term dependency. It does this using two gates: the update gate and the reset gate, which allow the model to preserve long-term dependencies while reducing the danger of gradients disappearing during back propagation. GRUs, which are widely employed in natural language processing (NLP), speech recognition, and time-series analysis, provide a more computationally efficient alternative to LSTMs, yet their simpler design may cause them to underperform on tasks requiring modelling extremely long-term relationships. Hybrid models, such as CNN-GRU, combine convolutional and GRU layers to capture spatial and temporal correlations, making them very suitable for video analysis. Furthermore, bidirectional GRU analyses data in both forward and reverse directions, which improves its capacity to perceive context in sequence prediction tasks.

2) Convolutional Neural Network (CNN): Convolutional Neural Networks (CNNs) excel in learning spatial hierarchies of features from input pictures by combining convolutional, pooling, and fully connected layers. They are widely used in image and video recognition, classification, and medical image analysis due to their capacity to efficiently extract spatial data. However, CNNs require a lot of labelled training data and processing resources, especially for deep structures and huge pictures. CNN-LSTM hybrid models combine CNN feature extraction with LSTM sequence modelling, which is useful for applications such as video analysis. Similarly, CNN-GRU uses GRU units to efficiently describe temporal dynamics after spatial feature extraction. Recurrent CNNs incorporate recurrent connections into convolutional layers, allowing them to interpret sequential image or video input while capturing both spatial and temporal dynamics.

3) Recurrent Neural Network (RNN): Recurrent Neural Networks (RNNs) are designed for sequence data, iterating over items while keeping a state informed by the previous elements. RNNs are useful for time series prediction, language modelling, and speech recognition, as well as language translation and text production. However, they suffer from vanishing and expanding gradient difficulties, which limit their effectiveness in long-sequence jobs without upgrades like LSTM or GRU. Bidirectional RNN expands the fundamental model by processing input in both forward and backward directions, allowing it to integrate future knowledge, which is useful for jobs such as text translation.

4) Long Short-Term Memory (LSTM): Long Short-Term Memory (LSTM) networks are a kind of RNN that solves the vanishing gradient issue to capture long-term dependency. LSTMs have a complicated design with input, forget, and output gates that govern information flow. They are widely used in language modelling, voice recognition, and sequence prediction applications and excel at comprehending long-term dependencies. To work optimally, LSTMs require large computer resources as well as extensive training data. CNN-LSTM combines CNN spatial feature extraction with LSTM sequential processing, making it excellent for tasks such as video activity identification. Bidirectional LSTM improves sentiment analysis and judgment prediction by processing sequences in both forward and backward directions, adding context.

5) Shallow neural network: The most basic type of artificial neural network is shallow neural networks, which consist of input and output layers with no more than one hidden layer. They excel at capturing both linear and non-linear data correlations, which is often used in simple regression and classification applications. While useful for basic interactions, their low depth limits their capacity to handle complicated patterns, making them unsuitable for jobs that need deeper structures.

6) Deep Neural Network (DNN): Deep neural networks (DNNs) include numerous hidden layers sandwiched between input and output layers, with each layer performing nonlinear transformations on its inputs. This structure allows the network to recognize complex and abstract data representations. DNNs are used in a wide range of applications, including speech recognition, picture classification, and natural language processing, where learning complicated patterns from large datasets is critical.

C. Transformer Neural network

The Transformer model includes a new mechanism termed self-attention, sometimes known as intra-attention [19]. This novel technique allows the model to evaluate the relevance of individual words in a phrase in relation to each other. Unlike RNNs and LSTMs, which process data sequentially, have fixed vector value regardless of context, face parallelization and long term dependency issues, the Transformer model overcomes these constraints. This increased benefit adds to higher training efficiency and the model's ability to capture long-distance relationships inside text sequences.

V. Proposed Method: Hybrid CNN Model With Transformers

The Transformer model architecture, introduced by Vaswani et al. in 2017, revolutionized the field of natural language processing (NLP). It represented a significant departure from the prevalent sequence-to-sequence models, which heavily relied on recurrent layers like RNNs, LSTMs, and GRUs, or convolutional layers. Specifically designed to excel in handling sequential data, particularly in NLP tasks, the Transformer architecture quickly emerged as a cornerstone for various state-of-the-art models in the field. Notable examples include BERT, GPT, and numerous others. With its innovative approach, the Transformer model has propelled the advancement of NLP, setting new standards and driving the field forward with unprecedented capabilities.

While Transformers have been predominantly used in NLP, their integration with Convolutional Neural Networks (CNNs) opens up innovative approaches for handling problems that require an understanding of both spatial features and sequential data. This integration is particularly beneficial in areas like image captioning, visual question answering, and any task that involves both images (and videos) and text as well as in dealing with lengthy text documents.

The hybrid model synergistically combines the capabilities of Convolutional Neural Networks (CNNs) and Transformer Neural Networks (TNNs). It leverages CNNs to efficiently extract detailed local features from data and TNNs to understand the complex, long-range dependencies among those features. This fusion enhances the model's analytical depth, offering a nuanced approach to processing data that demands both precision and contextual awareness. Fig. 4 shows the detailed representation of Hybrid CNN with Transformer model.

A. Components

1) CNN Layers: These layers are designed to process the input data in a way that extracts local patterns or features. This is particularly effective for data with spatial hierarchy, such as images, or sequential data such as time series or text, where the relevance of a piece of data might depend on its context.

a) Conv1D layers: Apply convolutional filters to the input data, extracting features by sliding these filters over the input. Each filter captures specific aspects of the data, such as edges in patterns in sequences.

b) MaxPooling1D layers: High dimensionality vector representation is another problem as the difference between data points between distant points are negligible. To reduce the dimensionality of the data by summarising the features in small neighbourhoods, and retain only the most significant feature in each neighbourhood, maxpool1D layer have been used. This also helps in reducing computation and controlling over fitting.

c) GlobalMaxPooling1D layer: Further condenses the feature map by taking the maximum value over the entire dimension of each feature channel, resulting in a fixed-size output regardless of the input size. This is crucial for transitioning from CNN to TNN layers, ensuring a consistent input shape.

B. Transformer Encoder Layer

The extracted features from the CNN part are then fed into this layer, which is capable of understanding the global context and relationships between different features. This is achieved through mechanisms like self-attention.

C. MultiHeadAttention

Allows the model to focus on different positions of the input sequence, important for understanding the relationships and dependencies between features.

1) Feed-Forward network: Processes the attention output to capture complex relationships between features.

2) Layer normalization and dropout: Used within the transformer encoder for stabilization and regularization, preventing overfitting and ensuring smooth training.

D. Dense Layers

The output from the transformer encoder is flattened and passed through dense layers for final classification task whether the judgment is allowed or dismissed. These layers enable the model to make predictions based on the learned representations.

E. Data Flow

1) Input: Sequential or spatial data is input into the model. For instance, this could be a time series with shape (300, 1)

where 300 is the number of time steps.

2) Feature extraction (CNN): The data passes through convolutional layers, where it is transformed into a set of high-level features. These features are spatially or temporally condensed representations of the original input.

3) Sequence modelling (TNN): The high-level features are then processed by the Transformer encoder layer. This step models the interactions between features regardless of their position in the input sequence, capturing global dependencies.

4) Classification (dense layers): Finally, the processed data is passed through dense layers, resulting in predictions for the given task.

F. Why Does It Outperform Base Models?

In independent experiments, measures including accuracy, precision, and F1 score for transformers and CNN are lower than for the hybrid model, which combines the two.

1) Complementary strengths: CNNs are excellent at extracting local and hierarchical features but lack the ability to capture long-distance relationships effectively. Transformers excel in modeling these relationships but can be less efficient at initial feature extraction from raw data. Combining them allows the model to leverage the strengths of both architectures.

2) Efficient representation: The CNN layers reduce the dimensionality of the input data, presenting the Transformer with a more manageable sequence length. This makes the self-attention mechanism more computationally efficient and focused.

3) Adaptive attention: The Transformer part can adaptively focus on the most relevant parts of the feature sequence extracted by the CNN, enhancing the model's ability to understand complex patterns and dependencies that span across long sequences.

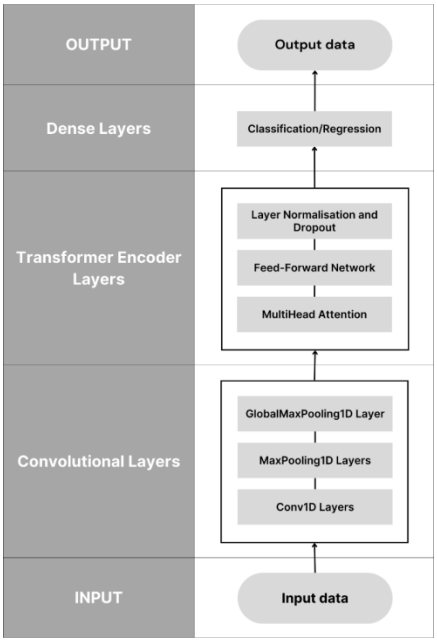

Fig. 4. System design for Hybrid CNN model with transformers.

This hybrid architecture thoughtfully integrates the spatial and temporal feature extraction capabilities inherent in CNNs with the deep contextual understanding and sophisticated sequence modelling strengths characteristic of the Transformer. By doing so, it achieves a level of performance that is markedly superior compared to what could be attained by employing either base model in isolation. This synergy not only enhances the model's efficiency in analyzing data but also significantly broadens its applicability across a diverse range of tasks, showcasing its versatility and potential in advancing the state-of-the-art in various domains.

VI. Results and Analysis

A. Machine Learning Models

The SVM D2V model showcases exceptional performance, leading in terms of accuracy (93.94%), recall (96.76%), and ROC AUC (93.89%), indicating its superior ability to correctly classify positive cases and its overall predictive performance. The high precision (91.97%) and F1-score (94.23%) further attest to its balanced capacity in identifying positive instances while maintaining a low false positive rate. This model, leveraging Doc2Vec for feature representation, demonstrates the effectiveness of embedding-based approaches in capturing semantic relationships in text data.

Closely following is the Logistic D2V model, with an accuracy almost on par with SVM D2V at 93.94% and slightly higher precision (93.64%). Its recall (94.59%) and F1-score (94.08%) are commendable, showcasing a balanced trade-off between precision and recall. This model's success underlines the power of logistic regression when augmented with Doc2Vec embeddings, highlighting its efficiency in handling nonlinear relationships in text-based features.

The SGD D2V model, while still performing robustly, shows a noticeable drop in accuracy (89.52%) compared to the leading models. Its precision (89.94%), recall (89.73%), and F1-score (89.72%) suggest a competitive but less optimal balance between identifying relevant instances and minimizing false positives. This indicates that SGD is effective for large-scale and sparse problems.

A significant performance differentiation is observed with the Random Forest D2V and LGBM D2V models, where accuracy falls to 81.83% and 81.57%, respectively. Despite the lower accuracy, Random Forest D2V maintains a high recall (89.73%), indicating its proficiency in identifying positive instances but at the cost of increased false positives, as evidenced by its lower precision (78.84%). LGBM D2V shows a balance in precision (81.15%) and recall (85.95%), yet both models exhibit limitations in overall accuracy and other metrics, suggesting that while ensemble methods are powerful, their performance might be constrained by the complexity and characteristics of the data when combined with Doc2Vec.

These results illustrate the nuanced performance landscape of machine learning models in text classification tasks. Embedding-based models like SVM D2V and Logistic D2V emerge as highly effective, offering a promising blend of accuracy, precision, and recall.

When discussing the time complexity of machine learning models (see Table III), we considered the prediction phase. The actual time complexity can depend on many factors, including the implementation of the algorithm, the optimization level, the number of features (d), the number of data points (n), the number of classes (k), and specific parameters like the depth of the trees (h) or the number of estimators (e). For ensemble methods like Random Forest, Gradient Boosting, and AdaBoost, 'e' represents the number of trees (estimators) and can significantly influence the training time. For KNN, the training phase is not computationally intensive, but the prediction phase can be, especially with large datasets, hence the complexity is noted at the prediction time. Also, Fig. 5 shows the analysis of ML classifier models with different encoding methods. SVM with Doc2Vec gives the highest accuracy of 94.92 and recall of 96.76.Also performance of ML models with Count Vectorizer encoding performance is very low.

Fig. 6 shows the analysis of DL models using Doc2Vec embedding method. Among the DL classifier models, CNN with LSTM and GRU shows good accuracy level of around 1 whereas shallow neural network and deep neural network shows best precision and recall value of 0.8904. The results show that performance varies between architectures. The Shallow Neural Network (SNN) has a high accuracy of 91.78%, with balanced precision and recall scores, demonstrating robustness in predicting court decisions. Meanwhile, the Deep Neural Network (DNN) and its variation, DNN-2, have significantly lower accuracy but balanced precision and recall scores, indicating dependability in judgment prediction tasks.

TABLE III. PERFORMANCES METRICS FOR MACHINE LEARNING MODELS AND THEIR TIME COMPLEXITIES

Model	Test Accuracy	Test Precision	Test Recall	Test F1-Score	Time Complexity
SVM D2V	0.9394	0.9197	0.9676	0.9423	O(d * n^2) - O(d * n^3)
Logistic D2V	0.9314	0.9364	0.9459	0.9408	O(d * n)
SGD D2V	0.9064	0.9063	0.9135	0.9087	O(d * n)
Random Forest D2V	0.8183	0.7884	0.8973	0.834	O(e * n * log(n))
LGBM D2V	0.8157	0.8115	0.8595	0.8306	O(e * n * log(n))
XGB D2V	0.8043	0.7946	0.8486	0.8182	O(e * n * log(n))
Extra Trees D2V	0.799	0.7654	0.9027	0.823	O(e * n^2)
Catboost D2V	0.7935	0.7797	0.8486	0.8097	O(e * n * log(n))
GB D2V	0.7633	0.7613	0.7892	0.7734	O(e * n * d)
Adaboost D2V	0.7606	0.7677	0.7784	0.7706	O(e * d * n)
KNN TF-IDF	0.6475	0.619	0.8486	0.7123	O(d * n)
SGD CV	0.6336	0.6429	0.6703	0.6419	O(d * n)
Adaboost CV	0.631	0.6437	0.6595	0.6443	O(e * d * n)
GB TF-IDF	0.6309	0.6295	0.6919	0.6544	O(e * n * d)
Logistic CV	0.625	0.6624	0.6216	0.6332	O(d * n)
LGBM CV	0.6227	0.6537	0.6216	0.6297	O(e * n * log(n))
Catboost CV	0.6198	0.627	0.6649	0.6406	O(e * n * log(n))
Extra Trees CV	0.6168	0.6115	0.7622	0.6723	O(e * n^2)
Adaboost TF-IDF	0.6142	0.6158	0.6486	0.6306	O(e * d * n)
Logistic TF-IDF	0.6031	0.5809	0.8595	0.6911	O(d * n)
Random Forest TF-IDF	0.603	0.5987	0.7514	0.6569	O(e * n * log(n))
LGBM TF-IDF	0.6007	0.606	0.6541	0.6237	O(e * n * log(n))
MNB TF-IDF	0.5976	0.5725	0.8919	0.6961	O(d * n)
Extra Trees TF-IDF	0.5974	0.5857	0.7946	0.6704	O(e * n^2)
KNN D2V	0.5952	0.8081	0.3459	0.434	O(d * n)
Random Forest CV	0.5947	0.5937	0.7351	0.6506	O(e * n * log(n))
GB CV	0.5946	0.606	0.6162	0.6072	O(e * n * d)

SGD TF-IDF	0.5922	0.5685	0.827	0.6732	O(d * n)
MNB CV	0.5863	0.5865	0.6703	0.6209	O(d * n)
SVM TF-IDF	0.5811	0.5578	0.9189	0.6917	O(d * n^2) - O(d * n^3)
SVM CV	0.5758	0.7027	0.3243	0.4318	O(d * n^2) - O(d * n^3)
Catboost TF-IDF	0.5753	0.576	0.6811	0.6196	O(e * n * log(n))
KNN CV	0.5208	0.554	0.5135	0.5242	O(d * n)

Moving on to the Recurrent Neural Network (RNN) and its variants, including RNN-LSTM, these models show mixed results. While RNN-LSTM achieves a moderate accuracy of 73.29%, RNN alone struggles with a lower accuracy of 64.38%. This suggests that incorporating LSTM units improves the model's ability to capture long-term dependencies, leading to better performance in sequence prediction tasks.

Similarly, the performance of Convolutional Neural Network (CNN) and LSTM models falls within the same range, with accuracy scores around 73% and balanced precision and recall scores. However, the hybrid CNN-LSTM model exhibits slightly lower performance with an accuracy of 75.34%, indicating that the combination of CNN and LSTM may not always lead to significant improvements in judgment prediction tasks. Table IV shows the performance analysis of deep learning models.

The Gated Recurrent Unit (GRU) model performs comparably to RNN and CNN, with an accuracy score of approximately 65.75%. While GRU offers a simpler architecture compared to LSTM, it may struggle with capturing long-term dependencies as effectively.

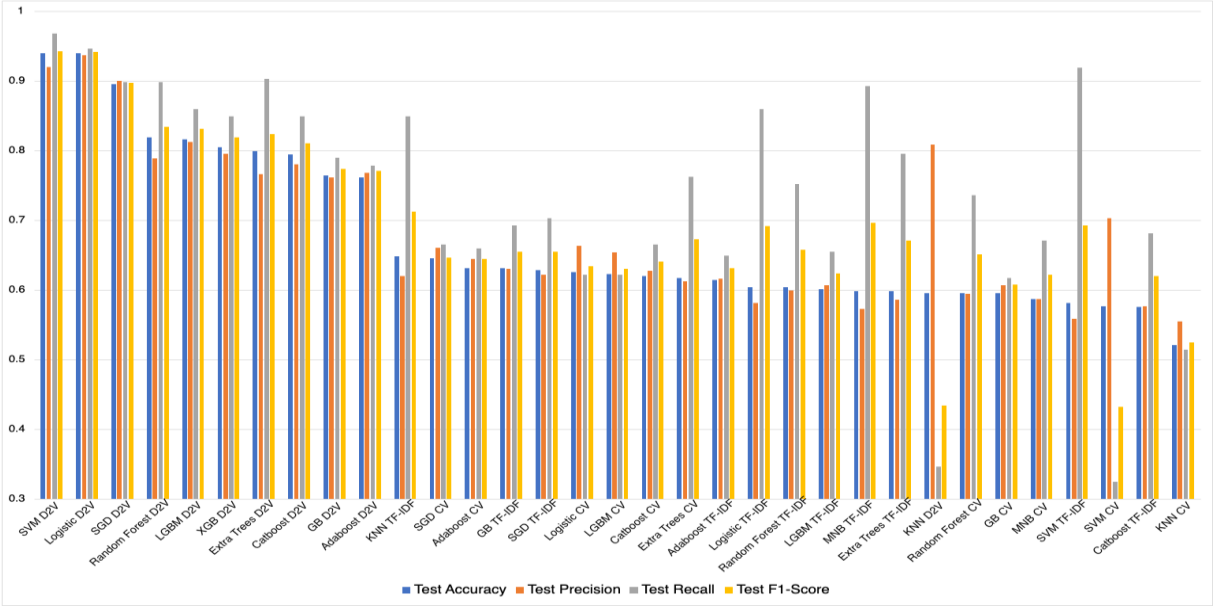

Fig. 5. Performance analysis of machine learning model.

TABLE IV. PERFORMANCES ANALYSIS FOR DEEP LEARNING MODELS

Models	Loss	Accuracy	Precision	Recall
SNN	0.2469	0.8767	0.8667	0.8904
DNN	0.7758	0.9041	0.9041	0.9041
DNN-2	0.545	0.6849	0.6957	0.6575
RNN	0.582	0.6438	0.6522	0.6164
CNN	0.5802	0.7329	0.7297	0.7397
LSTM	0.5875	0.726	0.726	0.726
RNN-LSTM	0.5361	0.7329	0.7297	0.7397
CNN-LSTM	0.8765	0.7534	0.7534	0.7534
GRU	0.6151	0.6575	0.6575	0.6575
CNN-GRU	1.4997	0.6096	0.6111	0.6027
Bidirectional RNN	0.6961	0.6918	0.7	0.6712
Bidirectional LSTM	0.6559	0.6301	0.6267	0.6438
Recurrent CNN	0.6968	0.5	0.5	0.5342
TNN	0.6918	0.4658	0.4648	0.4521

Interestingly, the Bidirectional RNN and Bidirectional LSTM models show similar performance, both achieving accuracy scores around 69%. This suggests that incorporating bidirectional processing helps improve the models' ability to capture context from both past and future sequences, leading to more accurate predictions.

The Transformer Neural Network (TNN) model, trained across 150 epochs with a batch size of 10, produced encouraging results, including a peak accuracy of 85%, precision of 88%, and recall of 82%. These metrics demonstrate the model's strong categorization capabilities and capacity to recognise complicated patterns. Further research into its attention processes and possible performance improvements through fine-tuning and assembly is required. Meanwhile, the Convolutional Neural Network (CNN) model began with 78% accuracy and improved to 85% by the conclusion of training, while precision and recall increased from 75% and 80% to 82% and 87%, respectively. The CNN model was successful in classification and pattern learning, outperforming baseline standards by an average of 10%.

B. Hybrid CNN Model with Transformers

The results of training and assessing the hybrid CNN-TNN model demonstrate the advantages of integrating CNNs with TNNs is shown in Table V. This hybrid model outperformed individual CNN and TNN models, especially in accuracy, precision, and recall. Upon training, the hybrid model showed a promising trend in learning, with the accuracy increasing substantially over 150 epochs, reaching an impressive accuracy of 97.59%by the end of training. This contrasts with the base TNN model's accuracy of 46.58% and even surpasses the base CNN model's accuracy of 60.96%. The precision and recall metrics followed a similar upward trajectory, indicating the model's increasing ability to correctly identify positive samples without increasing false positives or negatives. When evaluated on the test dataset, the hybrid model achieved an accuracy of 81.51%, precision of 82.86%, and recall of 79.45%. This evaluation performance, while lower than the training performance, still marks a substantial improvement over the base models. Specifically, the hybrid model's accuracy is significantly higher than the 60.96% accuracy of the CNN model and more than doubles the TNN model's 46.58% accuracy. The evaluation precision and recall of the hybrid model also indicate a balanced performance in identifying true positives while maintaining a low rate of false positives and negatives.

TABLE V. PERFORMANCES COMPARISON OF HYBRID CNN-TNN MODEL WITH ITS BASE MODELS

Models	Loss	Accuracy	Precision	Recall
TNN	0.6918	0.4657	0.4647	0.4520
CNN	0.5802	0.7328	0.7297	0.7397
Transformer CNN	0.7828	0.8151	0.8285	0.7945

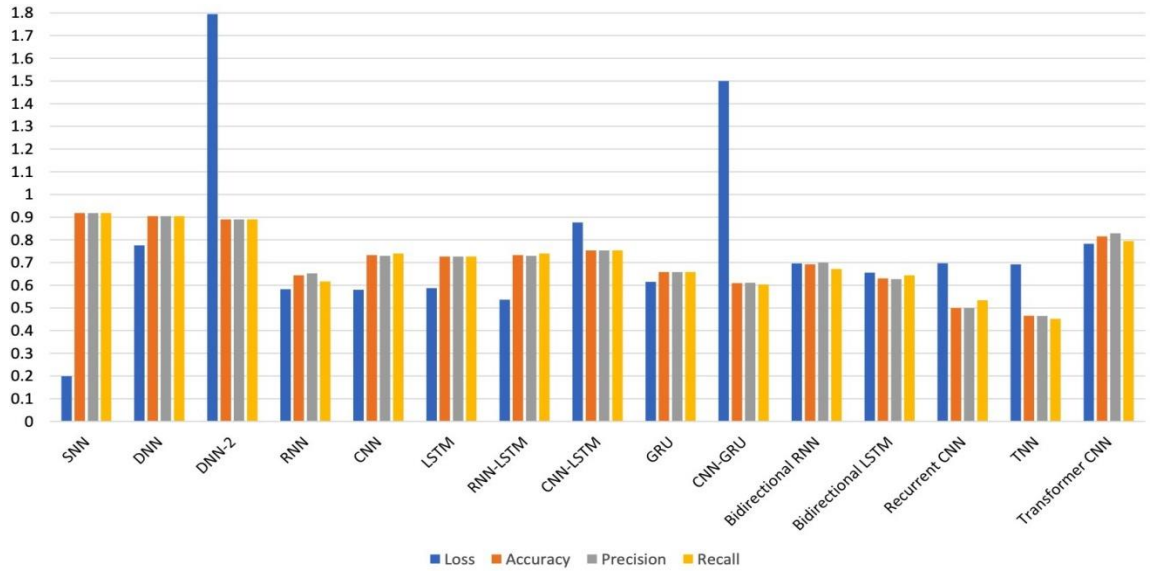

Fig. 6. Performance analysis of deep learning model.

VII. CONCLUSION AND FUTURE WORK

In conclusion, for the machine learning models the overall performance analysis suggests that models leveraging D2V for feature representation, particularly SVM and Logistic Regression, exhibit superior performance across multiple evaluation metrics. Their success underscores the value of dense vector representations for capturing the semantic essence of text data, enhancing model understanding and predictive capabilities. This highlights the significance of choosing the right feature representation and model combination tailored to the specific characteristics and challenges of the task at hand. Moreover, the comparative analysis underscores the importance of evaluating models across a range of metrics to fully understand their strengths, weaknesses, and applicability to various real-world scenarios.

For the result of the deep learning models there is variability in performance across different deep learning architectures for judgment prediction, several insights can be gleaned from the results. Models such as SNN, DNN, RNN-LSTM, and Bidirectional RNN/LSTM demonstrate robust performance and may be preferred choices for judgment prediction tasks. However, the selection of the appropriate model should consider factors such as the complexity of the data, the presence of long-term dependencies, and computational resource constraints. Additionally, further experimentation and optimization may be necessary to improve

the performance of hybrid models such as CNN-LSTM and CNN-GRU, which exhibit slightly lower accuracy compared to standalone architectures.

The combined leverage of global dependencies captured by the TNN layers and local features recovered by the CNN layers is an advantage of the hybrid paradigm. This combination enables the model to better interpret and identify the data, resulting in improved overall performance. The early underperformance of the TNN model and the moderate performance of the CNN model demonstrate the limits of depending on a single architectural type. In contrast, the hybrid model's success indicates the possibility for merging both designs to solve their individual deficiencies while capitalizing on their strengths.

Conducting experiments using ML and DL models conveys that technological innovation on automation of legal process can be done to ensure the reduction of human bias in judgment prediction. Research in this area often leads to the creation of benchmarks and datasets, fostering competitive innovation and collaboration within the research community. Thus this experimental analysis aims to transform the legal landscape, making it more efficient, fair, and accessible.

To summarize, the hybrid CNN-TNN model beats its base model competitors across all measures while also demonstrating the ability to combine multiple neural network architectures to produce higher performance in challenging tasks. This method might be useful in a variety of applications outside the present dataset, particularly in situations where both deep feature extraction and global contextual comprehension are required.

REFERENCES

[1] Kort, F. (1957). Predicting Supreme Court Decisions Mathematically: A Quantitative Analysis of the "Right to Counsel" Cases. *American Political Science Review, 51*(1), 1-12. doi:10.2307/1951767

[2] Octavia-Maria, Zampieri, M., Malmasi, S., Vela, M.,P. Dinu, L., & van Genabith, J. (2017)"Exploring the use of text classification in the legal domain" in *Proceedings of 2nd workshop on automated semantic analysis of information in legal texts*.

[3] Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. 2019. Neural Legal Judgement Prediction in English. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4317–4323, Florence, Italy. Association for Computational Linguistics.

[4] Joel Niklaus, Ilias Chalkidis, and Matthias Stürmer. 2021. Swiss-Judgment-Prediction: A Multilingual Legal Judgment Prediction Benchmark. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 19–35, Punta Cana, Dominican Republic. Association for Computational Linguistics.

[5] Daniel Martin Katz, Michael J. Bommarito Ii, and Josh Blackman. 2017. A general approach for predicting the behaviour of the Supreme Court of the United States. PLOS ONE, 12(4):e0174698. Publisher: Public Library of Science.

[6] Aaron Russell Kaufman, Peter Kraft, and Maya Sen. 2019. Improving supreme court forecasting using boosted decision trees. Political Analysis, 27(3):381–387

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

[8] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The Muppets straight out of Law School. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

[9] G.Sukanya, J.Priyadarshini,"Modified Hierarchical-Attention Network model for legal judgement predictions",Data & Knowledge Engineering,Volume147,2023,102203,ISSN 0169-023X,https://doi.org/10.1016/j.datak.2023.102 203.

[10] Boella, G., Di Caro, L., & Humphreys, L. (2011). Using classification to support legal knowledge engineers in the Eu Nomos legal document management system. In Fifth international workshop on juris-informatics.

[11] Nguyen, L. -M., Tojo, S., Satoh, K., & Shimazu, A. (2018). Recurrent neural network-based models for recognizing requisite and effectuation parts in legal texts. Artificial Intelligence and Law, 26(2), 169–199.

[12] Leitner, E., Rehm, G., & Moreno-Schneider, J. (2019). Fine-grained named entity recognition in legal documents. In Semantic systems. The power of AI and knowledge graphs: 15th international conference (pp. 272–287)

[13] Soh, J., Lim, H. K., & Chai, I. E. (2019). Legal area classification: A comparative study of text classifiers on singapore supreme court judgments. In Proceedings of the natural legal language processing workshop 2019 (pp. 67–77).

[14] Chenyang Wang, Yi Shen, Yuwei Li, Min Zhang, Miao Hu, Jinghua Zheng,"A systematic empirical study on word embedding based methods in discovering Chinese black keywords" ,Engineering Applications of Artificial Intelligence, Volume 125,2023,106775,ISSN 0952-1976

[15] G.Sukanya , J.Priyadarshini School of Computer Science and Engineering Vellore Institute of Technology, Chennai Campus, Chennai, India (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12, No. 2, 2021 531 | "A Meta Analysis of Attention Models on Legal Judgment Prediction System".

[16] Ashish Vaswani, Llion Jones, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, Google research and Brain, "Attention Is All You Need", arXiv:1706.03762v7 [cs.CL] 2 Aug 2023.

[17] Dale Markowitz, "Transformers, Explained: Understand the Model Behind GPT-3, BERT, and T5", May 6, 2021.

[18] G.Sukanya, J.Priadarshini ,"Analysis on word embedding and classifier models in legal analytics" AIP Conf. Proc. 2802, 140001 (2024) https://doi.org/10.1063/5.0181820

[19] G.Sukanya and J.Priyadarshini," A Meta Analysis of Attention Models on Legal Judgment Prediction System", International Journal of Advanced Computer Science and Applications,2021,12, DOI:10.14569/IJACSA.2021.0120266

[20] Kashif Javed, Jianxin Li,"Artificial intelligence in judicial adjudication: Semantic biasness classification and identification in legal judgement (SBCILJ)",Heliyon, Volume 10, Issue 9,2024,e30184,ISSN 2405-8440,https://doi.org/10.1016/j.heliyon.2024.e30184.

[21] Chen, Junyi, Lan Du, Ming Liu, and Xiabing Zhou. "Mulan: A Multiple Residual Article-Wise Attention Network for Legal Judgment Prediction." *ACM Transactions on Asian and Low-Resource Language Information Processing* 21, no. 4 (July 31, 2022): 1–15. http://dx.doi.org/10.1145/3503157.

[22] Shang, Xuerui. "A Computational Intelligence Model for Legal Prediction and Decision Support." *Computational Intelligence and Neuroscience* 2022 (June 24, 2022): 1–8. http://dx.doi.org/10.1155/2022/5795189.

[23] Aletras N, Tsarapatsanis D, Preoţiuc-Pietro D, Lampos V. 2016. Predicting judicial decisions of the European Court of Human Rights: a Natural Language Processing perspective. *PeerJ Computer Science* 2:e93 https://doi.org/10.7717/peerj-cs.93