International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 03 | Mar 2025 www.irjet.net p-ISSN: 2395-0072

LEGAL TEXT CLASSIFICATION AND TEXT ANALYSIS

T Sudhir Babu [1], A Tanooj Sai Kumar [2], B Siddhartha Reddy [3], B Rahul [4], K Raja [5], K Jogeeshwar Singh [6],

[1]*Professor, Dept of AI&DS, VVIT, Andhra Pradesh, India*
[2]*Student, Dept of AI&DS, VVIT, Andhra Pradesh, India*
[3]*Student, Dept of AI&DS, VVIT, Andhra Pradesh, India*
[4]*Student, Dept of AI&DS, VVIT, Andhra Pradesh, India*
[5]*Student, Dept of AI&DS, VVIT, Andhra Pradesh, India*
[6]*Student, Dept of AI&DS, VVIT, Andhra Pradesh, India*

Abstract - *Our research introduces a Legal Text Classification System aimed at automating document categorization in legal management. By leveraging advanced NLP techniques, the system efficiently categorizes legal documents into predefined classes, streamlining retrieval processes. We review complexities in legal document processing and advancements in text classification, employing various algorithms like LSTM, SVM, and Random Forest. Utilizing tools such as Python, scikit-learn, and TensorFlow, along with data preprocessing and deep learning architectures, our project delivers a robust NLP model and intuitive interface. Our evaluation methodology includes model performance metrics, comparative analysis, and user feedback integration, contributing to NLP advancements and offering a valuable tool for legal professionals.*

Keywords— NLP Techniques, LSTM, SVM, Random Forest Text Classification, optimizer, Legal Documents, Document Categorization, Evaluation Metrics

1. INTRODUCTION

In the realm of legal document management, the project "Legal Text Classification for Document categorization" aims to tackle the intricate challenge of categorizing documents into predefined classes. This endeavor is driven by the pressing need for efficient and accurate methods to organize and retrieve vast repositories of legal texts. Leveraging advanced Natural Language Processing (NLP) techniques, our project endeavors to develop a robust system capable of automating document categorization processes with unprecedented precision. The approach integrates a variety of machine learning algorithms, including Support Vector Machines (SVM), Random Forest, and sophisticated deep learning architectures such as Long Short-Term Memory (LSTM) networks. Through this comprehensive approach, our project seeks to revolutionize document management practices in legal domains, offering significant improvements in efficiency, accuracy, and overall productivity for legal professionals and practitioners.

1.1. Overview of Legal Text Classification:

With the rapid growth of digital legal documents, legal professionals face challenges in organizing, retrieving, and analyzing vast amounts of textual information. Legal Text Classification and Text Analysis aims to leverage Natural Language Processing (NLP) and Machine Learning (ML) techniques to automate and enhance the processing of legal texts.

This project focuses on classifying legal documents based on categories such as case law, contracts, regulations, or legal opinions. Additionally, it involves text analysis to extract insights, detect entities (e.g., parties, dates, legal references), and perform sentiment or contextual analysis.

The legal system relies heavily on precedent, were past judicial decisions influence future rulings. However, analyzing vast amounts of case law manually is time-consuming and complex. Case Law Prediction using Text Classification and Analysis aims to leverage Natural Language Processing (NLP) and Machine Learning (ML) to predict case outcomes based on legal texts, improving decision-making for legal professionals.

1.2 Importance of Text Analysis In Law:

Legal text analysis plays a crucial role in modern law by leveraging Natural Language Processing (NLP) and Machine Learning (ML) to process, classify, and interpret large volumes of legal documents. Here are some key reasons why legal text analysis is essential:

- **Efficient Legal Research**: Lawyers and researchers often need to analyze vast amounts of case law, statutes, and contracts.

- Automated text analysis helps quickly identify relevant precedents, legal principles, and critical arguments.

- **Improved Case Law Prediction**: By analyzing past judicial decisions, AI models can predict case outcomes based on similar legal arguments.

- This helps lawyers assess the probability of success in court and strategize accordingly.

- **Contract Analysis and Compliance**: Legal text analysis helps in detecting risk clauses, obligations, and non-compliance in contracts.

- Companies can use AI to ensure that agreements meet regulatory requirements.

- **Enhanced Access to Justice**: Automated legal text processing allows individuals and small firms to access legal insights without expensive legal fees.

- AI-driven legal chatbots provide guidance based on past legal cases and statutes.

- **Sentiment and Bias Detection in Judgments**: Legal text analysis can identify patterns of judicial bias or trends in rulings based on demographics, case types, or jurisdiction.

- This promotes fairness and transparency in the legal system.

- **Speeding Up Document Review & Due Diligence**: AI-powered document review tools assist law firms in quickly summarizing and analyzing lengthy contracts and court rulings.

- Saves time and costs in legal proceedings and corporate mergers.

- **Legal Knowledge Management**: Large legal databases are difficult to navigate manually. Text classification and topic modeling help in organizing legal knowledge efficiently.

- **Regulatory Compliance Monitoring**: Companies can use automated text analysis to monitor legal and regulatory updates and ensure ongoing compliance with the law.

1.3 Objectives of Project:

The primary objective of this project is to utilize Natural Language Processing (NLP) and Machine Learning (ML) to analyze and classify legal texts, with a particular focus on case law prediction. By automating legal text processing, the project aims to enhance legal research, improve case outcome forecasting, and support decision-making for legal professionals.

One key goal is to develop an automated system for case classification, enabling the categorization of legal documents based on case type (e.g., civil vs. criminal cases) and judicial outcomes (e.g., granted vs. denied appeals). Additionally, the project seeks to build predictive models that analyze legal arguments, precedent cases, and judicial reasoning to estimate the likelihood of case rulings. To achieve this, NLP techniques such as tokenization, lemmatization, Named Entity Recognition (NER), and TF-IDF will be employed for legal text preprocessing and feature extraction, ensuring that key entities like judges, laws cited, plaintiffs, and legal arguments are accurately identified.

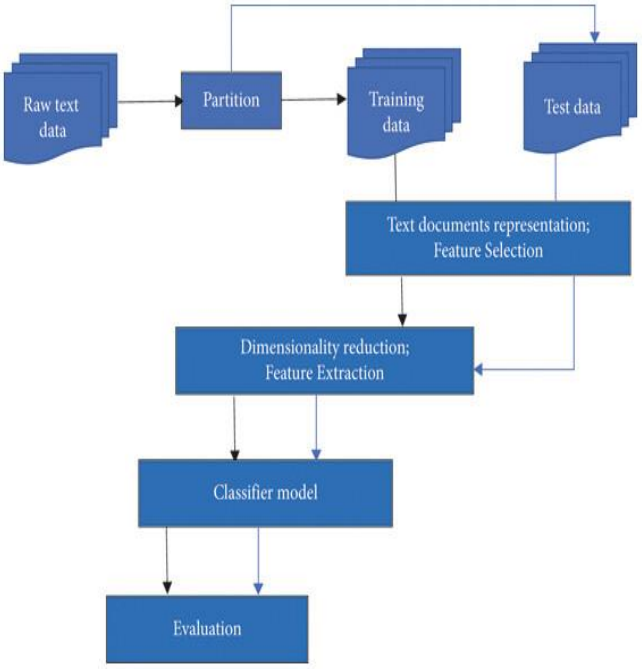

Fig 1.3 ARCHITECTURE OF TEXT CLASSIFICATION

2. PROPOSED IDEA

2.1 Introduction:

In the modern landscape of legal practice, the exponential surge in digital documentation poses a significant obstacle for legal practitioners in effectively managing and analyzing extensive collections of legal texts. Our research project proposes the development of an automated system that utilizes cutting-edge computer techniques to streamline the categorization of legal documents. Through the integration of advanced natural language processing (NLP) methods and machine learning algorithms, our system seeks to redefine conventional document management methodologies within the legal sphere.

2.2 Data Cleaning:

In the proposed project for automating the sorting of legal documents, the initial step involves cleaning and preprocessing the raw text data to ensure its quality and

suitability for further analysis. The following outlines the data cleaning process implemented.

- **Handling Missing Values**: In any dataset, missing values can compromise the integrity of the analysis. Therefore, it's crucial to address them appropriately. In the context of legal text classification, rows with missing values in essential columns, such as "case text" containing the actual text of the legal document and "case outcome" specifying the outcome or category, are identified and removed. This ensures that the data set is used for the training and evaluation of the consistent for handling the values.

- **Converting Text to Strings**: Text data in its raw form may have various representations, such as integers, floats, or objects. To ensure uniformity and compatibility with subsequent processing steps, such as tokenization and feature extraction, the text data is converted into a string format. This conversion simplifies data manipulation and analysis, allowing for seamless integration with NLP techniques and machine learning algorithms.

- **Data Chunking and Concatenation**: Large datasets, particularly in the legal domain where documents can be extensive, may pose challenges in terms of memory and computational resources. To overcome this limitation, the dataset is read in smaller, manageable chunks. Each chunk is then processed independently, and the results are concatenated to form a cohesive dataset. This approach enables efficient handling of large-scale data without overwhelming system resources, ensuring scalability and performance.

- **Dropping Unnamed Columns**: Datasets obtained from various sources may contain columns with generic or irrelevant names, often labeled as 'Unnamed.' These columns 2 typically do not contribute meaningfully to the analysis and may contain redundant or extraneous information. As part of the data cleaning process, such columns are identified and removed from the dataset. By eliminating unnecessary columns, the dataset is streamlined, reducing clutter and improving clarity for subsequent analysis steps.

2.3 Tokenization:

It is a pivotal step in natural language processing (NLP), which is executed using the NLTK library to segment raw text into individual tokens, typically words or punctuation marks. NLTK, a widely adopted Python library, provides robust tools for NLP tasks. Through tokenization, NLTK dissects text based on predefined rules, like whitespace and punctuation, ensuring accurate token extraction. Tokenization in law classification is a fundamental preprocessing step that involves breaking down legal documents into smaller units, such as words, sentences, or sub words, to facilitate text analysis and classification.

2.4 Lemmatization:

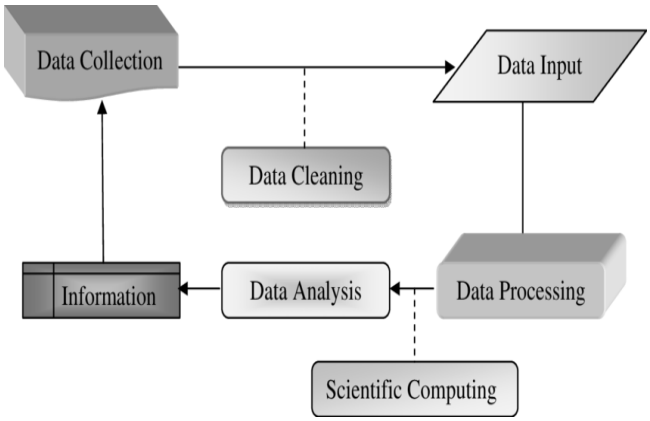

Fig 2.1 DATA CLEANING!

Lemmatization, a pivotal preprocessing step in natural language processing (NLP), is seamlessly executed using the NLTK library to transform words into their base or root forms, known as lemmas. This process ensures linguistic accuracy by considering a word's context and grammatical role within a sentence. By reducing words to their canonical forms, lemmatization aids in standardizing vocabulary, mitigating the impact of morphological variations, and enhancing the coherence of textual analyses. Leveraging NLTK's extensive lexical resources and linguistic rules, lemmatization facilitates effective information retrieval and text comprehension in legal documents.

2.5 Model Selection:

In our project focused on natural language processing (NLP) for legal document management, we have carefully selected and implemented NLP. Specific models tailored to the intricacies of legal text analysis. Here's an overview of the models we've integrated

- **Bidirectional Long Short-Term Memory:** Bidirectional Long Short-Term Memory (BiLSTM) is a neural network architecture designed to capture bidirectional dependencies in sequential data, making it particularly effective for understanding the contextual nuances present in legal texts. By processing legal documents as sequences of words or tokens, BiLSTM can learn intricate patterns and semantic relationships,

facilitating accurate classification into predefined categories. Its ability to model long-range dependencies and contextually rich information enables BiLSTM to excel in tasks requiring deep semantic understanding, such as legal document classification.

- **Support Vector Machine (SVM) with Text Features:** Support Vector Machine (SVM), when coupled with NLP-specific text feature extraction methods like TF-IDF or word embeddings, serves as a robust tool for text classification in the legal domain. By transforming legal texts into numerical representations using TF-IDF or embeddings, SVM can effectively delineate documents into distinct categories based on their semantic similarities. SVM with text features offers an interpretable and computationally efficient approach to document classification, complementing the deep learning capabilities of models like BiLSTM.

- **Random Forest Classifier with NLP Features**: The Random Forest algorithm, integrated with NLP-specific features extracted from legal texts, provides a versatile and powerful approach to document classification. By leveraging an ensemble of decision trees, Random Forest can effectively capture the complex relationships, and nonlinear patterns present in legal documents. When combined with features derived from NLP techniques such as TF-IDF or word embeddings, Random Forest offers a robust framework for accurately categorizing legal texts into predefined classes.

Through the integration of these NLP models, our project aims to develop a sophisticated system tailored specifically for legal document classification. By leveraging the unique capabilities of NLP techniques, we endeavor to streamline document management processes, empower legal professionals with efficient analysis tools, and facilitate expedited access to relevant legal information.

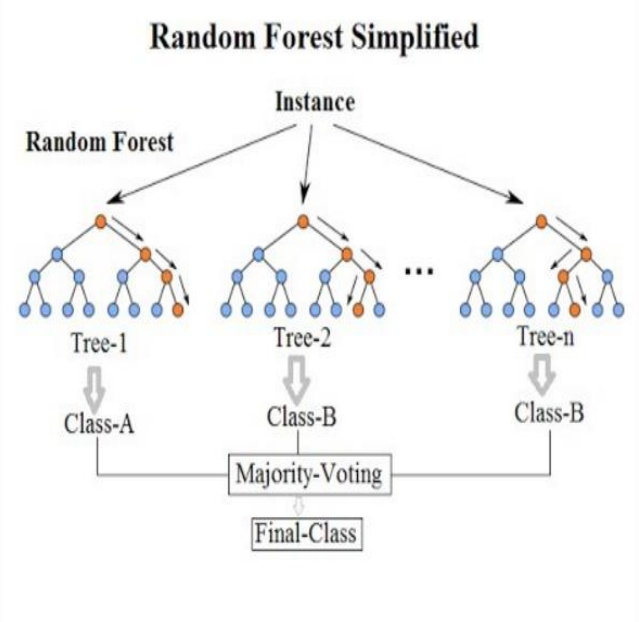

Fig 2.5 RANDOM FOREST ALOGRITHM

- **Ensemble Methods with NLP Models:** Ensemble methods, such as Random Forest or Gradient Boosting, synergistically combine multiple NLP models or integrate them with traditional machine learning algorithms to enhance classification performance. Leveraging the diversity of NLP models within an ensemble framework enables robust handling of the intricate linguistic structures and variations present in legal texts. By aggregating predictions from diverse models, ensemble methods offer improved generalization and robustness, contributing to more accurate and reliable document classification outcomes. Ensemble methods in NLP enhance legal text classification by combining multiple models to improve accuracy, robustness, and generalization. Techniques such as bagging, boosting, stacking, and voting help capture intricate linguistic structures in legal documents. websites. The deployment of this feature ensures that users can continuously evaluate their career readiness in real-time.

3. DATA COLLECTION AND PREPROCESSING

3.1 Dataset:

https://www.kaggle.com/datasets/shivamb/legal-citation-text-classification

The dataset from Kaggle provides structured information on legal cases, including case IDs, outcomes, titles, and full-text content. It is particularly useful for training and evaluating NLP models for legal citation classification. The case-id uniquely identifies each legal

case, while case-outcome specifies whether the case was cited in subsequent legal proceedings. The case-title gives a descriptive name for each case, and case-text contains the full legal document, including facts, arguments, and judicial decisions.

3.2 Data Processing:

Data processing is a crucial step in preparing the dataset for legal text classification. To ensure uniformity in data representation, the 'case-text' column is explicitly converted to string type, standardizing the format for NLP tasks. Additionally, handling missing values is essential for maintaining data integrity and reliability. Rows containing missing values in critical columns such as 'case-text' and 'case-outcome' are removed to prevent inconsistencies and improve model performance. These preprocessing steps help create a clean and structured dataset, ensuring accurate and efficient classification of legal documents. Additionally, vectorization techniques such as TF-IDF or word embeddings (e.g., Word2Vec, GloVe) transform textual data into numerical representations, making it suitable for machine learning models.

3.3 Data Analysis:

Data analysis involves exploring the dataset to extract meaningful insights before model training. One key technique is word cloud generation, where a subset of the dataset is sampled to visualize the most frequently occurring terms in legal documents. This visualization helps identify common themes, recurring legal terminology, and key topics within the dataset. By analyzing word frequency distributions, legal professionals and data scientists can better understand the dominant concepts in legal texts, which aids in feature selection and model optimization for classification tasks**.** Document length distribution analysis is performed using a histogram plot to examine the variation in legal text lengths across the dataset. This visualization helps in understanding the frequency and range of document sizes, providing insights into the complexity and diversity of legal texts.

3.4 Text Processing:

Text processing plays a vital role in preparing legal documents for classification, with tokenization being one of the key steps. Using the NLTK library's word-token function, the text is broken down into individual words or tokens, enabling more effective analysis and processing. Tokenization helps in structuring unstructured text data by segmenting sentences into meaningful units, which can then be used for further NLP tasks such as lemmatization, stop word removal, and feature extraction. This step is essential for improving the performance of machine learning models by providing a well-defined input representation.

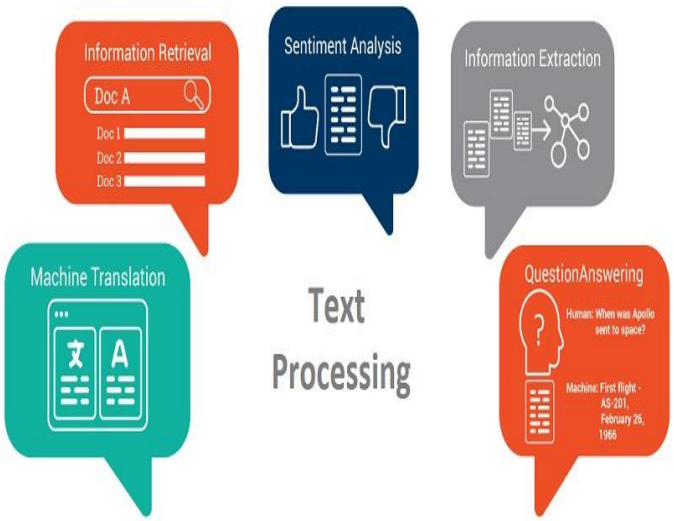

Fig 3.4 TEXT PROCESSING

4. MODEL TRAINING AND OPTIMIZATION

4.1 Model Initialization:

The TextClassifier class forms the foundation for implementing a Bidirectional LSTM (BiLSTM) model designed for text classification tasks. It is structured to provide flexibility and customization, allowing adaptation to different datasets and classification needs. The model begins with an embedding layer, which transforms tokenized text into dense vector representations, enabling the model to capture word semantics effectively Diagram.

A bidirectional LSTM layer follows, processing input sequences in both forward and backward directions to enhance contextual understanding. Fully connected layers further refine extracted features and map them to classification labels.

To prevent overfitting, dropout and regularization techniques are applied, ensuring model generalization. The classification process is optimized using an appropriate loss function, such as cross-entropy loss, combined with optimizers like Adam or RMSprop. By incorporating these components, the TextClassifier enables efficient and accurate legal text classification while allowing customization in embeddings, LSTM layers, and hyperparameters to improve performance.

Input, Embedding, Hidden, and Output Dimensions

Upon initialization, the TextClassifier class requires specifications for hidden, and output dimensions, allowing users to define the architecture based on their specific requirements.

- **Input Dimension**: Specifies the dimensionality of the input data, typically representing the vocabulary size or the number of unique tokens in the input text.
- **Embedding Dimension**: Determines the size of the dense embedding space where words or tokens are represented, playing a crucial role in capturing semantic relationships between words in the input data.
- **Hidden Dimension**: Defines the size of the hidden state vector in the LSTM units, impacting the model's ability to capture temporal dependencies and extract relevant features from sequential data.
- **Output Dimension**: Represents the number of output classes or categories in the classification task, enabling the model to predict the distribution of probability over predefined classes.
- **Bidirectional LSTM Architecture**: The TextClassifier class implements a Bidirectional LSTM (BiLSTM) model, consisting of two LSTM layers that process input sequences in both forward and backward directions. This bidirectional processing enables the model to capture contextual information from both past and future tokens within the input sequence, enhancing its ability to understand and represent the underlying semantics of the text effectively. By leveraging BiLSTM, the model improves its comprehension of complex linguistic patterns, making it well-suited for text classification tasks, particularly in domains like legal document analysis, where contextual meaning is crucial. Bidirectional processing enables the model to capture contextual information from both past and future tokens within the input sequence, significantly enhancing its ability to comprehend and represent the underlying semantics of the text.
- **Flexibility and Customization**: The TextClassifier class ensures flexibility and customization by allowing users to define key parameters such as input, embedding, hidden, and output dimensions. This adaptability enables the model to be tailored for various text classification tasks, including legal document analysis. The input dimension represents the vocabulary size, while the embedding dimension determines the quality of word representations. The hidden dimension controls the complexity of the LSTM units, influencing the model's ability to capture sequential dependencies. Users can experiment with different values for input, embedding, hidden, and output dimensions to optimize model performance, accommodate varying dataset characteristics, and adapt to specific classification tasks. Adjusting the input dimension allows the model to handle diverse vocabulary sizes, while modifying the embedding dimension helps in capturing richer semantic relationships.
- **Custom Dataset Creation**: A custom dataset class, CustomDataset, is implemented to manage input data for training while adhering to PyTorch's Dataset interface. This ensures seamless compatibility with PyTorch's training pipeline, allowing efficient data loading and processing. The CustomDataset class is responsible for handling text preprocessing, tokenization, and label encoding, converting raw legal documents into structured tensor representations suitable for model training. By defining this class, users can efficiently manage large datasets, apply necessary transformations, and enable optimized data batching, enhancing the overall performance of the BiLSTM-based text classification model.
- **Training Loop**: The model is trained over a defined number of epochs, optimizing with the Adam optimizer and cross-entropy loss function to improve classification performance. The training loop iterates over batches of data from the DataLoader, ensuring efficient processing of large datasets. In each iteration, the model forwards the input through the network, computes the loss, and performs backpropagation to adjust the model's parameters.

4.2 Interactive Text Classification Interface (SVM):

- **Data Loading and Preprocessing**: Like the LSTM approach, the legal dataset is loaded and preprocessed to ensure consistency in data preparation for the model. This involves tokenization, stopword removal, lemmatization, and vectorization using techniques like TF-IDF or word embeddings. Missing values are handled, and the text is converted into a structured numerical format suitable for input into the BiLSTM model. Standardizing preprocessing steps across different models ensures fair comparisons and optimal performance in legal text classification tasks.
- **TF-IDF Vectorization**: The preprocessed text data is transformed using TF-IDF (Term Frequency-Inverse Document Frequency)

vectorization, converting it into a numerical representation suitable for Support Vector Machine (SVM) classification. This transformation helps quantify the importance of words in a document relative to the entire dataset, enhancing the model's ability to distinguish between different legal text categories.

4.3 Random Forest Classifier Implementation:

A sample legal text is input into the trained Random Forest model to predict its classification outcome. The model's performance is evaluated using key metrics such as F1-score, precision, and accuracy, which provide insights into its effectiveness and reliability in legal text classification.

These evaluation metrics help assess how well the model balances precision and recall, ensuring it accurately categorizes legal documents. Additionally, comparing the Random Forest model's performance with other models (e.g., SVM, BiLSTM) facilitates informed decision-making and optimal model selection for real-world legal applications.

Once the Random Forest model has been trained on the TF-IDF transformed legal text data, it is tested using sample legal texts to assess its classification accuracy. The model predicts the appropriate category for a given legal document, determining whether the classification aligns with the actual label.

4.4 Model Hyperparameters:

Model hyperparameters play a crucial role in determining the behaviour and performance of the trained model. The learning rate controls the step size for updating model parameters during training, influencing convergence speed and stability.

The dropout probability helps prevent overfitting by randomly deactivating a fraction of neurons during training, enhancing the model's generalization ability. The batch size specifies the number of samples processed in each iteration, impacting computational efficiency and gradient updates.

The number of epochs defines how many times the entire dataset is passed through the model, affecting how well the model learns from the data. Additionally, the optimizer, such as Adam or SGD, determines how model parameters are updated to minimize loss and improve accuracy. The selection and fine-tuning of these hyperparameters are essential, often requiring multiple experiments to achieve optimal model performance for the specific legal text classification task.

4.5 Feature Extraction:

Feature extraction is a crucial step in preparing text data for machine learning models, as it converts unstructured text into a numerical format that algorithms can process effectively. One of the most widely used techniques for this purpose is TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization, which transforms pre-processed text into a structured numerical feature matrix.

In this approach, each document in the dataset is represented as a row, while each unique word in the

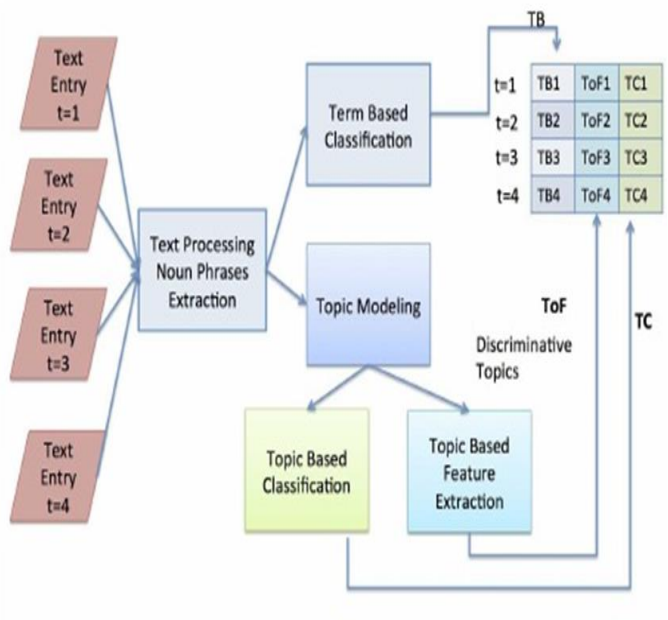

Fig 4.5 TEXT-BASED FEATURE EXRACTION

corpus forms a column. The values within this matrix represent the TF-IDF score of each word in a given document, which is computed by measuring how frequently a word appears in a document (Term Frequency) and adjusting it based on how rare or common the word is across the entire dataset (Inverse Document Frequency).

In legal text classification, extracting meaningful features from unstructured text is crucial for building effective machine learning models. Several feature extraction techniques help transform legal documents into numerical representations, enabling algorithms to process and classify them accurately.

One of the most used methods is TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization, which assigns weight to words based on their importance in a document relative to their occurrence across the entire dataset. This helps highlight domain-specific legal terms while reducing the impact of commonly used words.

5.SIMULATION-RESULTS

5.1 Evaluation Metrics for Model Performance:

Precision evaluates the proportion of correctly predicted positive instances among all predicted positives, which is essential in minimizing false classifications that could mislead legal interpretations. Recall assesses the model's ability to identify all actual positive cases, ensuring that no relevant legal documents are missing, which is particularly important in legal analysis. F1-score, as the harmonic mean of precision and recall, provides a balanced measure, helping to minimize both false positives and false negatives for improved classification reliability.

- **Accuracy:** Accuracy represents the proportion of correctly classified instances among the total number of instances evaluated, providing an overall measure of the model's correctness in predicting the class labels of legal documents. It is calculated as the ratio of correctly predicted cases to the total cases in the dataset. While accuracy is a fundamental metric for evaluating model performance, it may not always be sufficient in cases where class distribution is imbalanced.

- **Precision:** Precision measures the ratio of correctly predicted positive instances to the total number of instances predicted as positive. It evaluates the model's ability to minimize false positives, ensuring that when a legal document is classified into a particular category, it is indeed relevant to that category.

- **Recall:** Recall, also known as sensitivity or true positive rate, quantifies the proportion of correctly predicted positive instances out of all actual positive instances. It evaluates the model's ability to capture all relevant legal documents within a given category, ensuring that no important cases are overlooked. A high recall score indicates that the model successfully identifies most of the relevant documents, reducing the number of false negatives.

- **F1-score**: F1-score is the harmonic means of precision and recall, providing a balanced measure of a model's performance by considering both metrics simultaneously. It is particularly useful when dealing with imbalanced datasets, where one class has significantly more instances than others. A high F1-score indicates that the model is performing well in both minimizing false positives (precision) and capturing all relevant instances (recall). Unlike accuracy, which can be misleading in imbalanced datasets, the F1-score ensures that the model is not biased toward majority classes.

5.2 Cross-Validation:

Cross-validation is a technique used to evaluate the generalization performance of a trained model and assess its robustness to unseen data. One of the most used methods is k-fold cross-validation, where the dataset is partitioned into k subsets (folds). The model is trained on k-1 folds and tested on the remaining fold, repeating this process k times, with each fold serving as the test set once.

This iterative approach ensures that the model is evaluated on multiple subsets of data, reducing the risk of overfitting and providing a more reliable estimate of its performance. In legal document classification, where datasets can be imbalanced or contain nuanced text variations, cross-validation helps ensure that the model is not overly dependent on a specific subset of data.

5.3 Approaches:

- **LSTM approach:** The LSTM model was trained for 10 epochs, during which a gradual decrease in loss was observed across successive epochs. This reduction in loss indicates that the model effectively learned patterns from the training data and continuously adjusted its parameters to minimize prediction errors. Each epoch involved forward propagation, where the model made predictions, and backpropagation through time (BPTT), which updated the model's weights using the chosen optimizer (e.g., Adam). The iterative nature of this process helped refine the model's understanding of legal text sequences, improving its ability to classify documents accurately. As the model trained, it captured important semantic relationships and contextual dependencies in legal documents, leading to better generalization for unseen data. Monitoring the loss trend ensured that the model was learning effectively while preventing overfitting, allowing for a robust classification system.

- **SVM approach:** A user-friendly interactive interface was developed using ipywidgets, allowing legal professionals to input legal text and obtain real-time predictions from the SVM model. This interface enhances user experience by providing an intuitive and accessible way to classify legal documents without requiring technical expertise. By integrating the classification model into an interactive tool, legal workflows can be streamlined, enabling faster document analysis and decision-making. The interface facilitates practical deployment in real-world legal applications, making it easier for users to automate document

categorization and retrieve relevant legal information efficiently.

- **Random Forest Classifier approach:** The Random Forest classifier was trained on TF-IDF transformed data, achieving a competitive F1 score of 0.77 on the test set. This strong performance highlights the classifier's ability to effectively analyze and categorize legal texts by leveraging an ensemble of decision trees. The model benefits from robust feature selection and redundancy reduction, making it highly resilient to overfitting while maintaining high classification accuracy. Its effectiveness in handling complex legal language and diverse document structures further reinforces its suitability for legal text classification tasks.

- **Comparative Analysis:** A comprehensive evaluation of performance metrics, including accuracy, precision, recall, and F1-score, was conducted across the LSTM, SVM, and Random Forest models to assess their effectiveness in legal text classification. While all three models exhibited distinct strengths, the SVM model emerged as the most effective, achieving the highest F1-score and demonstrating superior precision in categorizing legal texts. This highlights its suitability for applications where precise outcome predictions are critical. Additionally, an analysis of model robustness, usability, and deployment feasibility provided valuable insights into the unique advantages of each model, ensuring that the best approach is selected based on specific legal classification requirements.

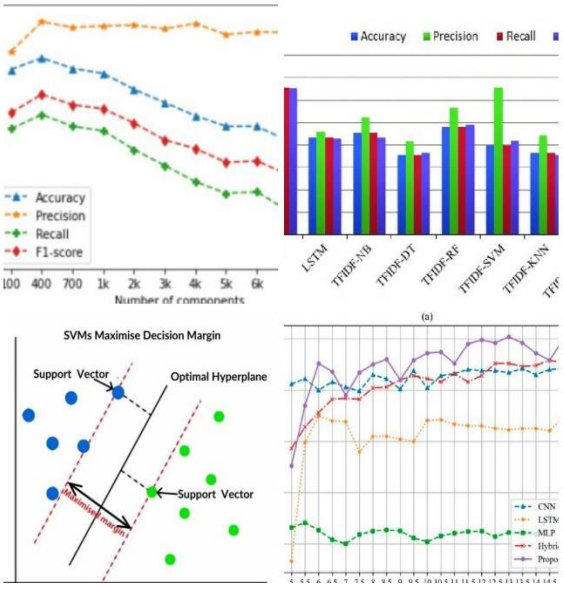

Fig 5.2 Approaches

6.CONCLUSION AND FUTURESCOPE

6.1 Conclusion:

Our research presents a sophisticated Legal Text Classification System aimed at automating document categorization in legal management. By leveraging advanced Natural Language Processing (NLP) techniques and machine learning algorithms such as LSTM, SVM, and Random Forest, the system efficiently classifies legal documents into predefined categories, significantly improving retrieval processes and workflow efficiency.

The integration of powerful tools like Python, scikit-learn, and TensorFlow, combined with comprehensive data preprocessing and deep learning architectures, has resulted in a robust NLP model. Additionally, the development of an intuitive user interface ensures seamless interaction, making legal text classification more accessible and effective for professionals in the field.

These efforts not only advance NLP applications but also provide a valuable tool for legal professionals, revolutionizing document management practices by significantly enhancing efficiency, accuracy, and productivity in the legal domain. Moving forward, continuous refinement and optimization, driven by user feedback and ongoing evaluations, will further improve system performance and usability.

6.2 Future scope:

Our Legal Text Classification System currently utilizes advanced NLP techniques such as tokenization, lemmatization, and TF-IDF vectorization to preprocess and extract meaningful features from legal documents. These preprocessing steps ensure that the text data is structured effectively for machine learning applications.

Following this, classification models such as LSTM, SVM, and Random Forest are trained on the processed data to categorize legal documents accurately. The LSTM model captures sequential dependencies in text, the SVM model efficiently classifies high-dimensional textual data, and the Random Forest model enhances classification through ensemble learning. By integrating these approaches, the system ensures reliable and efficient document categorization, facilitating streamlined legal research and management.

In the future, we aim to explore more sophisticated NLP methods such as BERT-based models for contextual understanding and transformer architectures for improved document representation.

7.REFERENCES

1. Legal Document Classification, [Online]. Available: https://ieeexplore.ieee.org/document/9207211

[Accessed: 10-Oct-2025].

2. Abid, A., Farrugia, N., and Yue, M., "Gradio: Build Machine Learning Web Apps in Minutes," [Online]. Available: https://gradio.app/

[Accessed: 05-Oct- 2025].

3. Streamlit, Inc., "Streamlit Documentation," [Online]. Available: https://docs.streamlit.io/

[Accessed: 08-Oct-2025].

4. Hugging Face, "huggingface_hub: Interact with HuggingFaceModels," [Online].Available: https://huggingface.co/docs/huggingface_hub/

[Accessed: 06-Oct- 2025].

5. Enhancing LSTM and Fusing Articles of Law for Legal Text Summarization [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-99-8181-6_9

[Accessed: 12-Oct-2025].