TERMS OF SERVICE FAIRNESS ANALYZER USING MACHINE LEARNING

Chris Mathew[1], Sushanth S Rao[2], Charan P[3], BVS Datta Goutham[4], Manohar R[5]

[1,2,3,4]*BE Students,* [5]*Assistant Professor, Department Information Science and Engineering,*

Sir M. Visvesvaraya Institute of Technology Hunasamaranahalli, Bengaluru-562157

ABSTRACT: ***Terms of Service (ToS) are fundamental factors in the creation of physical as well as online legally relevant relationships. Very often, they disregard the consumer protection law. In this perspective, a relevant issue is that public agencies in charge of control concretely lack the resources needed to effectively fight against such unlawful practices. We propose a definition of ToS unfairness and a novel machine learning-based approach to classify clauses in ToS, represented by using sentence embedding, into both categories and fairness classes. We use Naïve Bayes Classifier to achieve the desired categorization and result. Terms of Service (ToS) are fundamental factors in the creation of physical as well as online legally relevant relationships. They not only define mutual rights and obligations but also inform users about contract key issues that, in online settings, span from liability limitations to data management and processing conditions. Despite their crucial role, however, ToS are often neglected by users that frequently accept without even reading what they agree upon, representing a critical issue when there exist potentially unfair clauses. To enhance users' awareness and uphold legal safeguards, we first propose a definition of ToS unfairness based on a novel unfairness measure computed counting the unfair clauses contained in a ToS, and therefore, weighted according to their direct impact on the customers concrete interests.***

Keywords: ***TOS, Machine Learning, Naïve Bayes, Fairness, Clauses***

I. INTRODUCTION

Terms of service (ToS) typically explains the purpose and scope of the agreement between the service provider and the user. The introduction may also provide a brief overview of the key terms and conditions that users must agree to before using a particular product or service. When you sign up for an account on an online platform, you are typically asked to agree to the platform's terms of service (ToS) agreement. These agreements are legal documents that outline the rules and guidelines for using the platform, as well as the platform's policies regarding user conduct, content ownership, and liability. The terms of service agreement usually contain several sections, including: Acceptance: This section outlines that by using the platform, you agree to be bound by the terms of service agreement. User Conduct: This section outlines the platform's rules and guidelines for user behaviour, including rules for posting content, engaging with other users, and following applicable laws and regulations. Content Ownership: This section outlines who owns the content posted on the platform, and how that content can be used and shared, Liability: This section outlines the platform's liability limitations, disclaimers, and indemnification provisions. It also outlines what happens if users violate the terms of service agreement and Termination: This section outlines the circumstances under which the platform may terminate your account, and the consequences of termination. It's important to read and understand the terms of service agreement before agreeing to it, as it is a legally binding document. Some platforms may include provisions that limit your rights as a user, such as waiving your right to sue or participate in a class action lawsuit.

II. MOTIVATION

Terms of service (ToS) is to establish a legal agreement between the service provider and the user of a product or service. ToS typically outline the rules and guidelines that users must agree to in order to use a particular product or service. This agreement protects the rights of both the service provider and the user and helps to prevent misunderstandings or conflicts that may arise. There are several key motivations behind ToS, including: Protecting Intellectual Property: ToS often contain provisions that protect the intellectual property rights of the service provider. This may include restrictions on the use of copyrighted material or trademarks. Defining User Conduct: ToS may outline the acceptable use of a product or service and prohibit certain types of conduct, such as harassment or hate speech. This helps to create a safe and respectful environment for all users. Limiting: This helps to protect the service provider from lawsuits or other legal actions. By outlining the rules and guidelines that users must follow, ToS help to create a clear understanding of the expectations and responsibilities of both the service provider and the user. This can help to prevent misunderstandings and promote trust between the two parties.

III. RELATED WORKS

Guarino, A., Lettieri, N., Malandrino, D. et al proposed a machine learning-based approach to identify unlawful practices in online terms of service: analysis, implementation and evaluation : Terms of Service (ToS) are fundamental factors in the

creation of physical as well as online legally relevant relationships. They not only define mutual rights and obligations but also inform users about contract key issues that, in online settings, span from liability limitations to data management.

Edda Leopold, Jorg Kindermann proposed a Text Categorization with Support Vector Machine which represent Texts in Input Space : Different mappings of frequencies to input space, and combine these maps for categorization. It also does the linguistic preprocessing

Dan Hendrycks, Collin Burns, Anya Chen, Spencer Ball proposed a CUAD: An Expert Annotated NLP Dataset for Legal Contract Review : CUAD was created with legal experts from The Atticus Project and consists over 13,000 annotations. The transformer models is found to have nascent performance.

Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean proposed a Efficient Estimation of Word Representations in Vector Space : The representations of words derived by various models on a collection of syntactic and semantic language tasks. High quality word vectors can be trained using lower computational complexity.

IV. METHODOLOGY

A. The area of expertise to be examined is in the understanding of the natural language. The domain of problem consists of all the potent unfair clauses. We achieve to address this through Natural Language Understanding (NLU), technique derived from Natural Language Processing (NLP). NLP stands for Natural Language Processing, which is a part of Computer Science, Human language, and Artificial Intelligence. It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation. Natural Language Understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles. NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

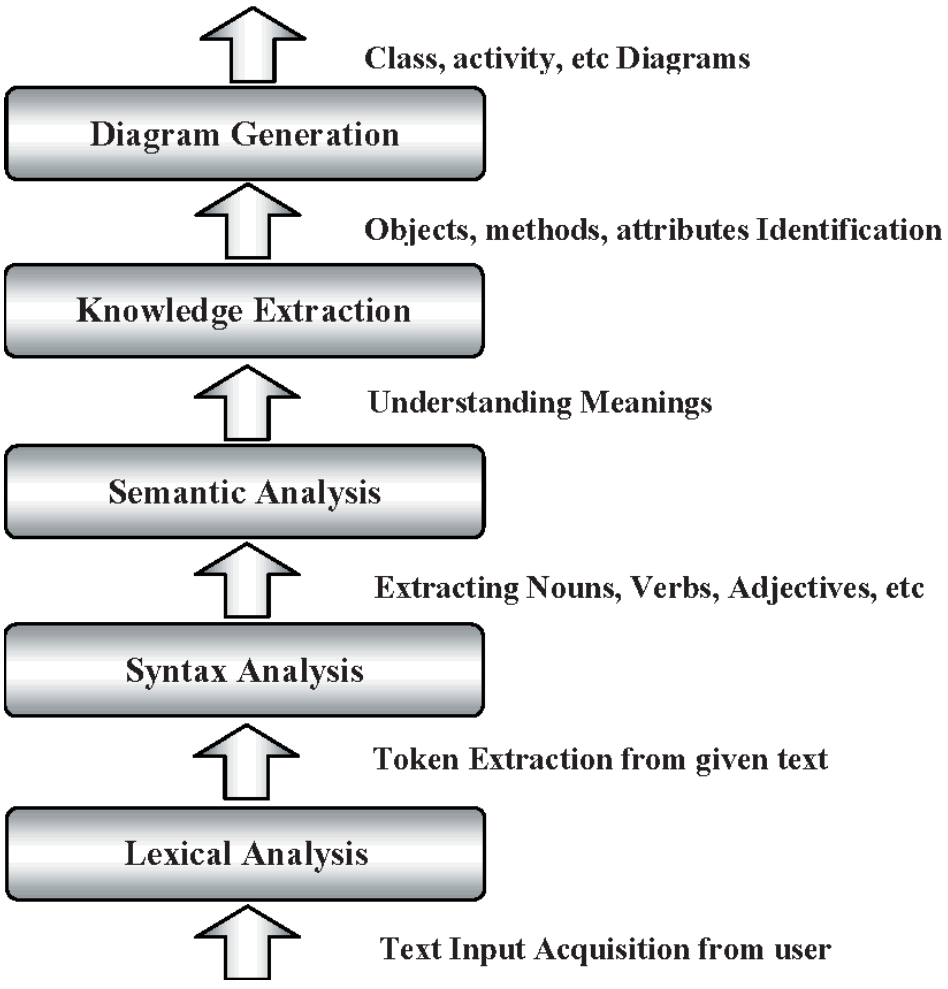

Fig 1. NLP map analyzing

B. In this section, we describe a novel machine learning-based method to classify ToS clauses according to pre-defined categories and a novel machine learning-based method to classify ToS clauses according to three fairness levels. We remark that to the best of our knowledge, this is the first work in which a machine learning-based method to classify ToS clauses according fairness level was proposed. To pursue this goal we defined a methodology encompassing three steps: Dataset Building: In this step, we download a set of JSON formatted resources, representing the ToS files containing the labeled clauses. Validation: The dataset has been split into training and testing sets. k-fold cross-validation was performed to validate different classifiers. Testing: The most used classifiers in the literature have been tested on the testing set with the best parameters found.

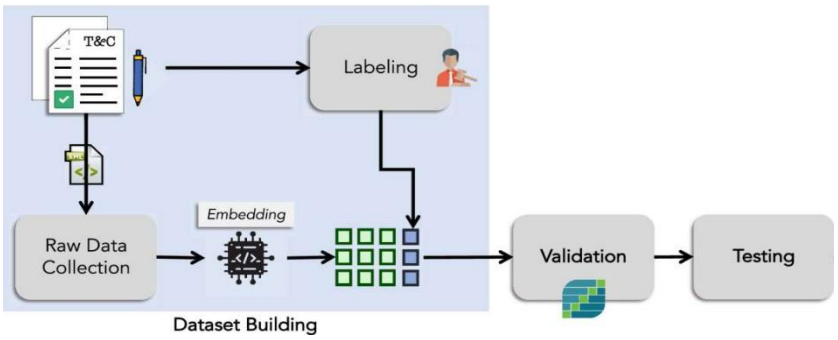

Fig 2. Architecture Overview

C. The workflow for the text-based classification model is shown below:

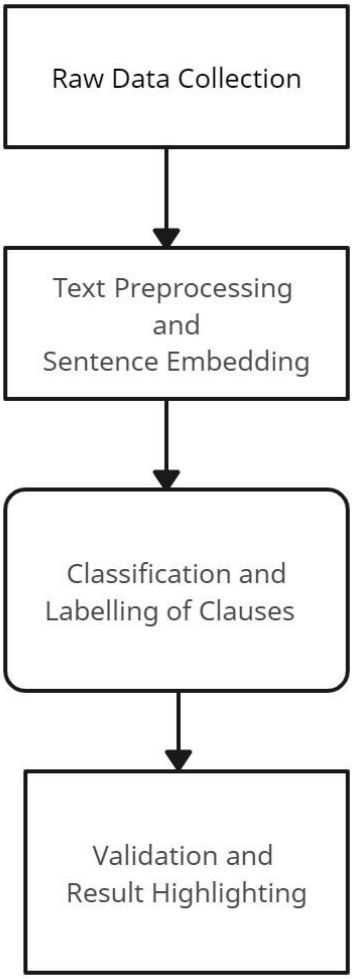

Fig 3. Model structure

D. Random Forest Module: Random Forest is a type of supervised learning algorithm that is used for classification, regression, and other tasks. It is an ensemble method that combines the predictions of multiple decision trees, which are constructed using random subsets of the training data and features. The idea behind random forests is to reduce overfitting and increase accuracy by using a combination of weak learners.

Random Forest works by creating a large number of decision trees and combining them to make a final prediction. Each decision tree in the forest is trained on a random subset of the training data and features. This helps to reduce the correlation between the trees and improve the accuracy of the final prediction. It is highly accurate and can handle both categorical and continuous data. It is resistant to overfitting, as it uses multiple decision trees instead of a single complex model. It is relatively easy to interpret and visualize the results.

Random Forest is a powerful machine learning algorithm that can be used for text classification tasks. It is a type of ensemble learning algorithm that constructs a forest of decision trees, where each tree is trained on a subset of the training data using a random selection of features. The final prediction is then made by aggregating the predictions of all the trees in the forest.

The first step in text classification using Random Forest is data preprocessing. This involves cleaning the raw text data and converting it into numerical features that can be used by the algorithm. Techniques such as tokenization, stop word removal, stemming or lemmatization, and feature scaling are used to transform the text data into a usable format.

The next step is feature extraction. After preprocessing the text data, relevant features are extracted that can be used to train the Random Forest algorithm. This can involve using techniques such as bag of words or TF-IDF to represent the text data as numerical features. Once the feature extraction is complete, we split the preprocessed data into a training set and a test set. The

training set is used to train the Random Forest algorithm, while the test set is used to evaluate the model's performance.

After splitting the data, the Random Forest algorithm is trained on the training set. Each decision tree in the forest is trained on a subset of the training set using a random selection of features. The algorithm then aggregates the predictions of all the trees in the forest to make a final prediction.

Once the model has been trained, we evaluate its performance on the test set using metrics such as accuracy, precision, recall, and F1 score. We can also use techniques such as cross-validation to further evaluate the model's performance. To fine-tune the Random Forest algorithm's performance, we can adjust its hyperparameters. This involves using techniques such as grid search or random search to find the optimal combination of hyperparameters that produce the best performance.

Finally, we can use the trained Random Forest algorithm to make predictions on new text data. This involves preprocessing the new text data and feeding it into the trained model to predict the class label of the new data.

Overall, Random Forest is a powerful and versatile approach that can achieve high accuracy for a wide range of text classification tasks. However, it can be computationally intensive and may require significant preprocessing and hyperparameter tuning to achieve optimal performance.

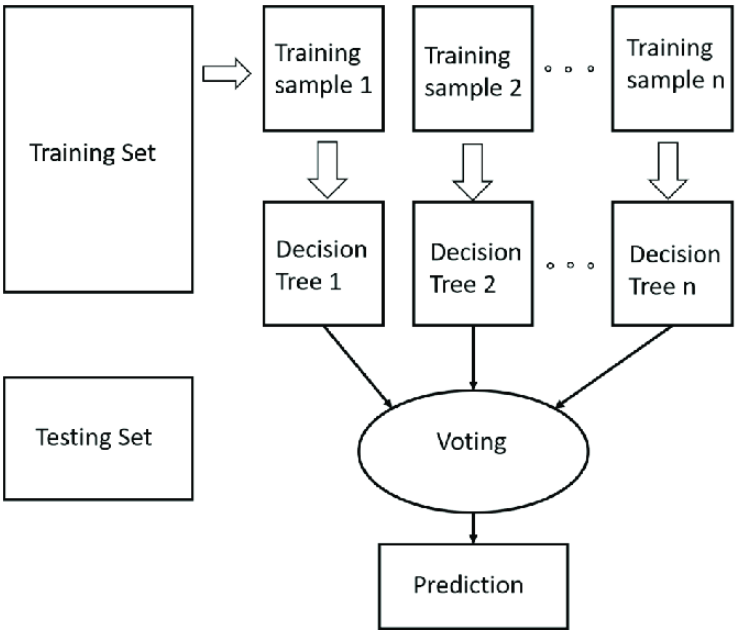

Fig 4. Random Forest based text classification flowchart

E. Support Vector Machine Module: Support Vector Machines (SVMs) are a type of supervised learning algorithm that is used for classification and regression tasks. SVMs are based on the idea of finding a hyperplane that separates the data into different classes. The hyperplane is chosen so that it maximizes the margin between the two classes.

SVMs work by mapping the input data into a high-dimensional feature space and then finding the hyperplane that separates the classes. This can be done using a kernel function, which computes the similarity between pairs of data points in the feature space. The key features of SVMs include that they are effective in high-dimensional spaces, making them suitable for text classification and other tasks. They are versatile and can handle both linear and non-linear data. They are relatively robust to outliers and noise in the data.

SVMs are a type of machine learning algorithm used for text classification tasks. The first step is to preprocess the text data by cleaning the text and converting it into numerical features that can be understood by the SVM algorithm. This involves techniques such as tokenization, stop word removal, and stemming or lemmatization. Next, the SVM algorithm is trained on the preprocessed text data. The algorithm works by finding the hyperplane that best separates the different classes in the feature space. This involves maximizing the margin between the hyperplane and the closest data points from each class. SVMs can use different types of kernels, such as linear, polynomial, or radial basis function (RBF), to map the data into the feature space.

After training the SVM model, it needs to be evaluated on a separate test set to measure its performance. This is typically done using metrics such as accuracy, precision, recall, and F1 score. If the model is not performing well, you can try adjusting the hyperparameters of the SVM algorithm, such as the kernel type, regularization parameter, and gamma. Once the model has been

trained and evaluated, it can be used to make predictions on new text data. This involves preprocessing the text data and feeding it into the SVM model to predict the class label of the new text data.

Overall, SVMs are a powerful algorithm for text classification tasks that can achieve high accuracy with careful preprocessing and hyperparameter tuning. It is important to choose an appropriate kernel function and regularization parameter to avoid overfitting the model to the training data.

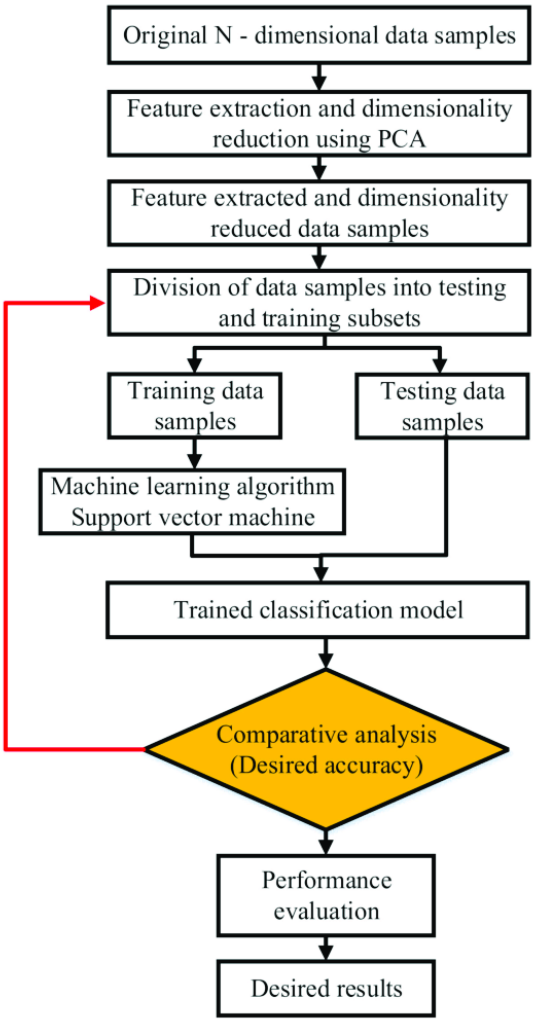

Fig 5. Support Vector Machine based text classification flowchart

F. Multinomial Naïve Bayes Module: Naive Bayes is a type of probabilistic machine learning algorithm used for text classification tasks. The algorithm works on the principle of Bayes' theorem, which states that the probability of a hypothesis (or class) given the observed evidence (or text document) is proportional to the probability of the evidence given the hypothesis multiplied by the prior probability of the hypothesis. In simpler terms, it calculates the probability of a text document belonging to a particular class based on the probability of each word in the document being associated with that class. Multinomial Naive Bayes is a variant of the Naive Bayes algorithm that assumes that the features are generated from a multinomial distribution, which is often the case in text classification tasks. The algorithm estimates the probabilities of each feature given each class label using a multinomial distribution and uses Bayes' theorem to calculate the probability of each class label given the input features.

To use Naive Bayes for text classification, we first preprocess the text data by cleaning the text and converting it into numerical features. This involves techniques such as tokenization, stop word removal, and stemming or lemmatization. Next, we split the preprocessed text data into training and test sets. The Naive Bayes algorithm is then trained on the training set by calculating the probabilities of each word occurring in each class. There are three types of Naive Bayes algorithms commonly used for text classification: Bernoulli Naive Bayes, Multinomial Naive Bayes, and Gaussian Naive Bayes. The choice of algorithm depends on the type of features used for text classification.

Once the Naive Bayes model has been trained, it needs to be evaluated on a separate test set to measure its performance. This is typically done using metrics such as accuracy, precision, recall, and F1 score. If the model is not performing well, you can try

adjusting the smoothing parameter, which helps to prevent zero probabilities when calculating the likelihood of each word in the test set.

The steps for text classification using Naive Bayes algorithm:

1. Data Preprocessing: This step involves cleaning the raw text data and converting it into numerical features that can be used by the algorithm. Text data may contain a lot of noise such as stop words, punctuations, and special characters that do not add any meaning to the text. Therefore, it is important to remove these noisy elements and normalize the text. Tokenization is a technique used to split the text into individual words or tokens. Stop words such as "the", "a", "an" can be removed as they do not provide any useful information. Stemming or lemmatization can be used to reduce words to their base form. Lastly, feature scaling can be performed to ensure that all features are on the same scale.

2. Feature Extraction: After preprocessing the text data, we need to extract relevant features that can be used to train the Naive Bayes algorithm. Bag of words is a popular technique used to represent the text data as numerical features. This involves creating a vocabulary of unique words that appear in the text data and then counting the frequency of each word in each document. TF-IDF is another technique that assigns weights to each word based on its importance in the document and its frequency in the corpus.

3. Train-Test Split: Once the feature extraction is complete, we split the preprocessed data into a training set and a test set. The training set is used to train the Naive Bayes algorithm, while the test set is used to evaluate the model's performance. A typical split is 80:20 or 70:30, where the larger portion of the data is used for training.

4. Model Training: We train the Naive Bayes algorithm on the training set by estimating the probability distribution of each feature for each class. This involves calculating the prior probabilities of each class and the likelihood of each feature given each class. The Naive Bayes algorithm assumes that the features are conditionally independent given the class label, which simplifies the computation and reduces the risk of overfitting.

5. Model Evaluation: Once the model has been trained, we evaluate its performance on the test set using metrics such as accuracy, precision, recall, and F1 score. Accuracy measures the proportion of correctly classified instances, while precision measures the proportion of true positives among the instances that were classified as positive. Recall measures the proportion of true positives among all the instances that actually belong to the positive class. F1 score is the harmonic mean of precision and recall and is used to balance the trade-off between them.

6. Hyperparameter Tuning: We can fine-tune the Naive Bayes algorithm's performance by adjusting its hyperparameters. This involves using techniques such as grid search or random search to find the optimal combination of hyperparameters that produce the best performance. The hyperparameters that can be tuned include the smoothing parameter alpha, the feature selection method, and the feature weighting method.

7. Prediction: Once the Naive Bayes algorithm has been trained and tuned, we can use it to make predictions on new text data. This involves preprocessing the new text data and feeding it into the trained model to predict the class label of the new data. The predicted class label can be used to make decisions or recommendations based on the text data.

Overall, text classification using Naive Bayes is a simple and effective approach that can achieve high accuracy for a wide range of text classification tasks. It is computationally efficient and requires minimal preprocessing and hyperparameter tuning. However, it assumes that the features are independent of each other, which may not always be the case in real-world text data.

After evaluating the Naive Bayes model, it can be used to make predictions on new text data. This involves preprocessing the text data and feeding it into the Naive Bayes model to predict the class label of the new text data. Overall, Naive Bayes is a simple yet powerful algorithm for text classification tasks that can achieve high accuracy with careful preprocessing and parameter tuning.

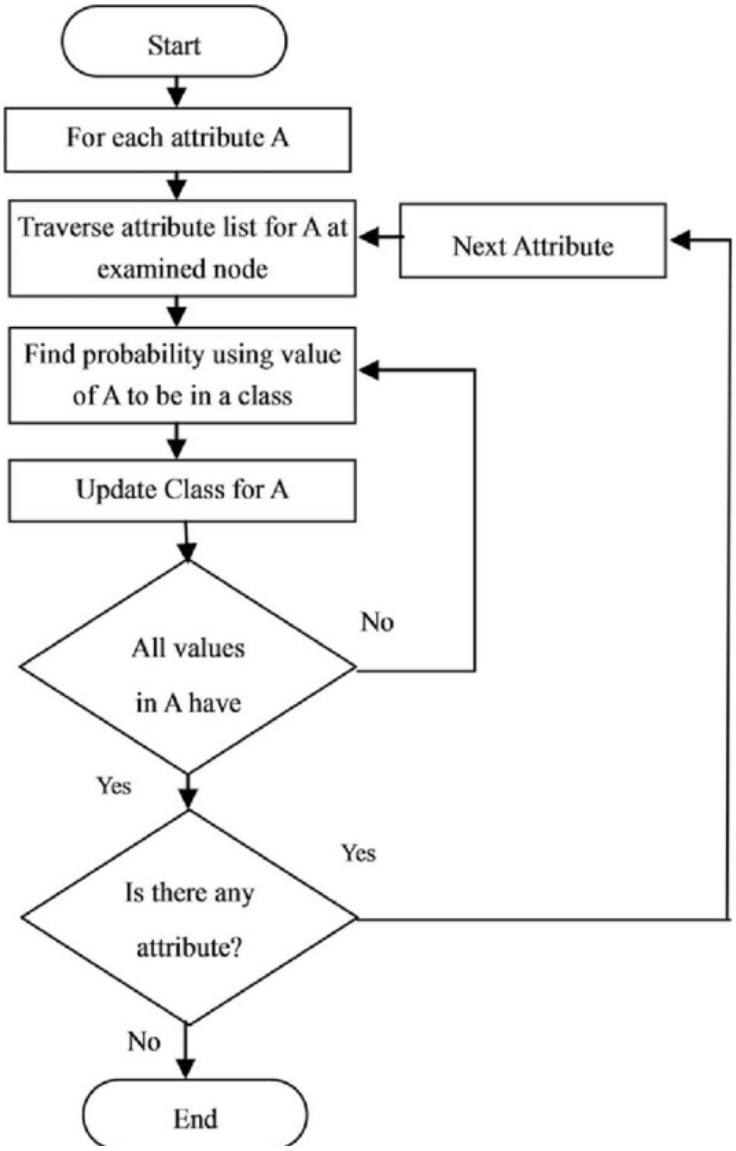

Fig 6. Naïve Bayes text classification flowchart

G. After considering and running all the modules in which we aimed to classify the clauses of terms and conditions agreements of platforms as fair or unfair. To achieve this, we experimented with the aforementioned three popular machine learning algorithms - Random Forest, Support Vector Machine (SVM), and Naive Bayes.

We evaluated the performance of each algorithm by using metrics such as accuracy, precision, recall, and F1 score. After comparing the results, we found that Naive Bayes achieved the optimal efficiency and accuracy for our task. Therefore, we have decided to proceed with Naive Bayes for our project.

The Naive Bayes algorithm works well for text classification tasks as it assumes that the features are conditionally independent given the class label. This simplifies the computation and reduces the risk of overfitting. Moreover, Naive Bayes is a simple and easy-to-implement algorithm that works well for large datasets. We are confident that the Naive Bayes algorithm will help us achieve our project's goal of classifying the clauses of terms and conditions agreements accurately and efficiently. We plan to further optimize the performance of the Naive Bayes algorithm by fine-tuning its hyperparameters and implementing feature engineering techniques.

Overall, we are satisfied with our experimentation and decision to proceed with Naive Bayes for our project. We believe that this algorithm will help us deliver a useful and valuable tool for users to evaluate the fairness of terms and conditions agreements of various platforms.

V. PROPOSED WORK

A. Proposed Algorithm is given below

Step 1: The Libraries Being Imported
Step 2: Fetch the Text or Clause from the User
Step 3: Preprocess the Text by removing punctuation, stop words
Step 4: Perform stemming and tokenize the resulting text from above step
Step 5: Vectorize the preprocessed text
Step 6: Classify the transformed input based on the trained model using Naïve Bayes
Step 7: Predict the label for the input using naïve bayes module
Step 8: Output the result to the user on the interface

B. Proposed Architecture and System Flow Diagram

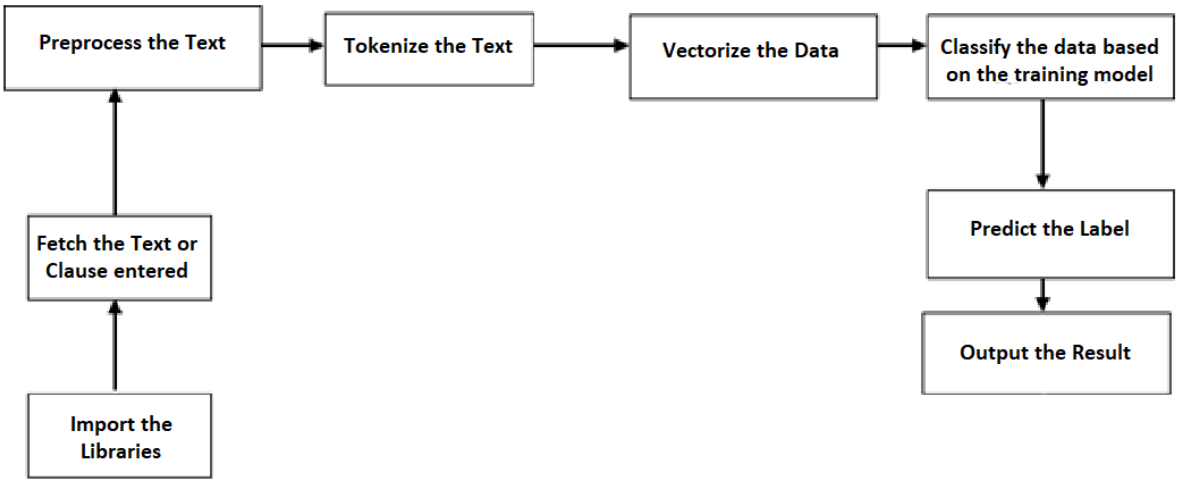

Fig 7. Overall Architecture

VI. EXPERIMENTAL RESULTS

A. Multinomial Naïve Bayes Overfitting Check for the Terms of Service Dataset

The graph represents the Multinomial Naïve Bayes Overfitting Check for the Terms of Service dataset. A orange line shows the testing accuracy, a blue line shows the training accuracy. After a considerable period of time with the increase in training set size, the projection approaches the similar metrics. As it learns more, the system will acquire more accuracy.

The x-axis shows the training set size, which is the smoothing parameter for the Multinomial Naive Bayes algorithm. The y-axis represents the accuracy of the algorithm on the training and testing datasets. We can see that with the increase in value of training set, the accuracy of the algorithm on the training dataset increases. The accuracy on the testing dataset don't decrease after a certain point. This indicates that the model is not overfitted on the training data and is able to generalize well on new data. The difference between the training and testing accuracies is minimum at the end of analysis.

Overall, this graph helps us understand the performance of the Multinomial Naïve Bayes algorithm on the Terms of Service dataset and provides insights on how to optimize its hyperparameters for better accuracy and generalization.

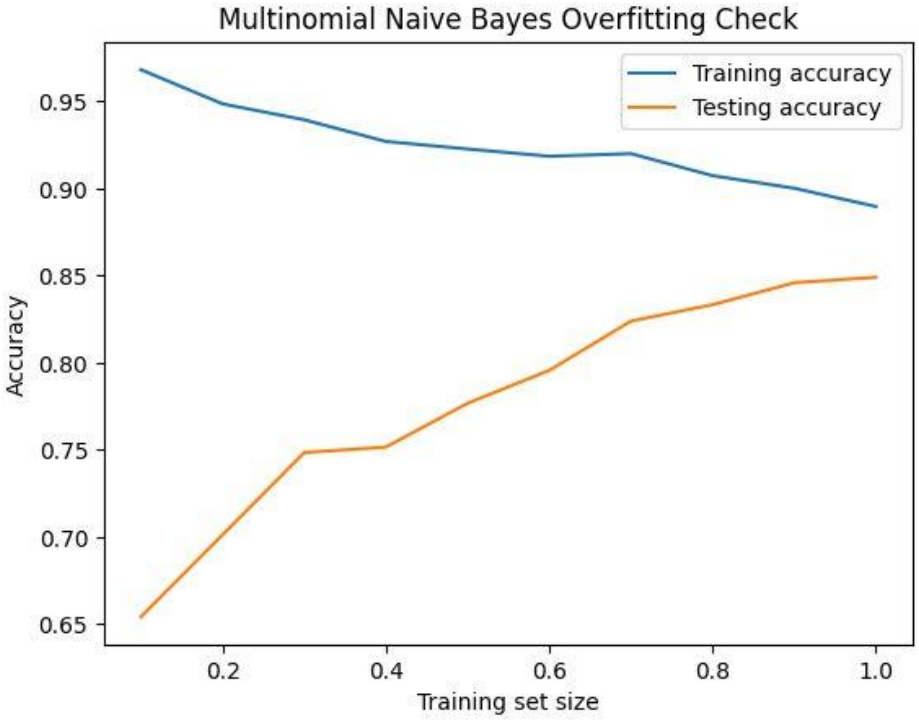

Fig 8: Training and Testing Accuracy Plot

B. Tuning the learning rate for efficient training

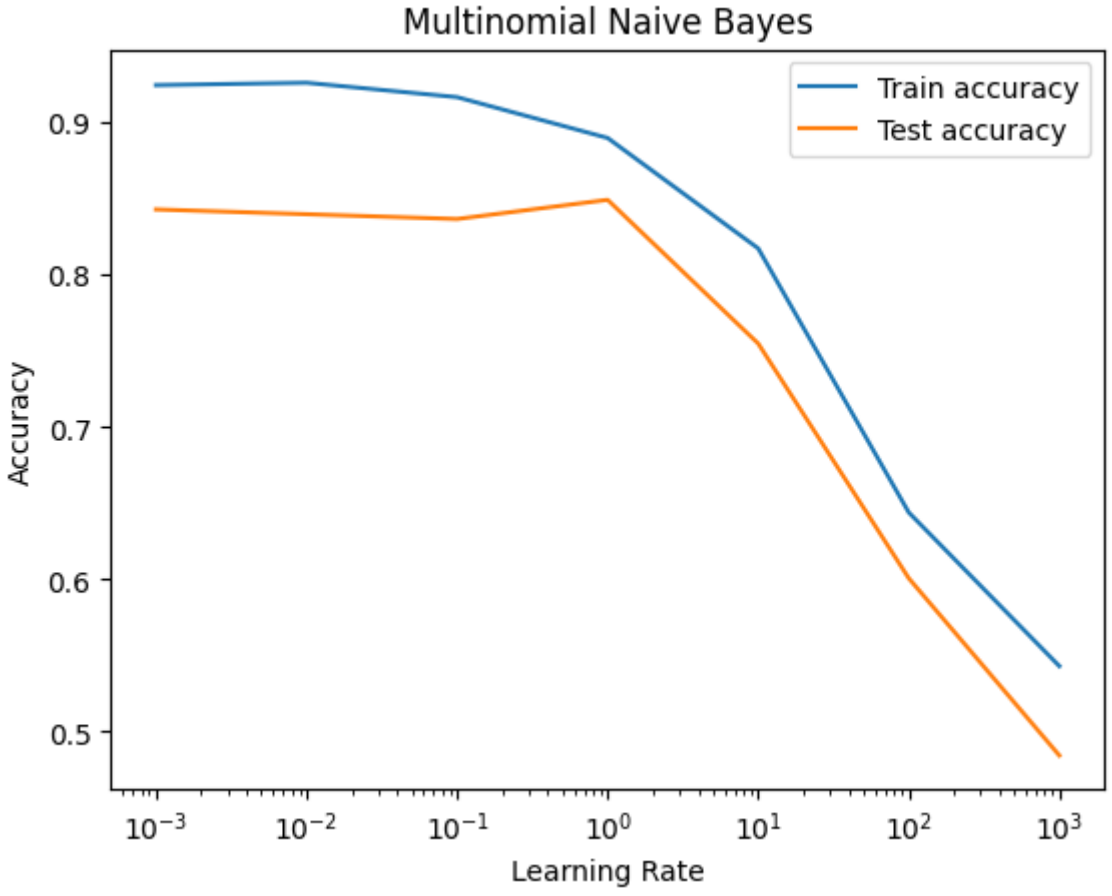

Fig 9: Plot describing change in accuracy with respect to learning rate

In Multinomial Naive Bayes, the smoothing parameter, learning rate, controls the strength of the smoothing applied to the probability estimates. A smaller value of learning rate means less smoothing, while a larger value means more smoothing. The graph of accuracy as a function of learning rate can help to determine the optimal and accurate value of learning rate for the given dataset and task. As you can see, the accuracy of the classifier increases as the value of learning rate increases up to a certain point, beyond which the accuracy starts to decrease due to over-smoothing.

The optimal value of learning rate is the one that maximizes the accuracy of the classifier on the validation set. In our use case for classification of unfair clauses, the optimal value of learning rate is around 0.001. This value is determined by the dataset and task. The optimal value of learning rate is estimated using our validation set and it avoid overfitting.

C. Evaluation Metrics

0.8490566037735849

	precision	recall	f1-score	support
0	0.88	0.82	0.85	165
1	0.82	0.88	0.85	153
accuracy			0.85	318
macro avg	0.85	0.85	0.85	318
weighted avg	0.85	0.85	0.85	318

Fig 9. Accuracy Score of the Classification

Figure 9 shows the model's classification accuracy. The experimental results show an accuracy of 84.9%. The categories 0 and 1 represent fair and unfair clauses respectively. The model is able to achieve the precision of 0.88 for the fair clauses and 0.82 for the unfair clauses. The recall which measures the completeness of positive predictions is 0.82 for fair clauses and 0.88 for unfair clauses. The F1-score, which measures the model's accuracy by taking the ratio of product of recall and precision over sum of recall and precision is 85%. The F1-score is same for fair and unfair clauses at 85% so our model predicts the clauses to corresponding categories accurately.

VII. CONCLUSION

The "notice and choice" legal regime used to rule the agreement on online ToS has shown severe flaws. Due to various reasons (intricacies in the texts, lack of legal skills), users struggle to grasp all the implications of clauses they are agreeing upon and often end up skipping reading them. This allows companies to take advantage of inscrutable and unfair contractual clauses that limit their liabilities or allow them to arbitrarily interrupt services at any time.

To tackle this issue, we proposed a novel machine learning-based approach whose main goal is to make ToS more readable/understandable in order to increase user awareness and "technologically enhance" consumer rights protection. Our approach exploited Multinomial Naïve Bayes for the clauses category classification task that achieved F1-score of 85%. With regard to the results of the experiments about the comparison with state-of-the-art works, although showing slight performance differences, our approach is able to reach the optimal F1-score that tells about the accuracy of classifications.

This project tackles the issue of unfair clauses present in the terms and service agreements and equips user with awareness. This

leaves future scope to extend the application to encompass various other legal proceedings. The dataset collected and trained in this project was mainly on the online service providers. However other legal bindings can be obtained that are used in designing contracts for actors, musicians, sports, employment offers, house rent and so forth agreements. Any unfair clauses can be incorporated to be identified.

REFERENCES

[1] Obar JA, Oeldorf-Hirsc: Ignoring the privacy policies and terms of service policies of social networking services (2020)
[2] Micklitz HW, Pałka P, Panagis Y: Digital control of unfair term of online services. J Consumer Policy (2017)
[3] Lettieri N, Altamura A, Malandrino D: The legal macroscope: experimenting with visual legal analytics (2016)
[4] Alfonso Guarino, Nicola Lettieri, Delfina Malandrino & Rocco Zaccagnino: A machine learning-based approach to identify unlawful practices in online terms of service (2021)
[5] Edda Leopold, Jorg Kindermann: Text Categorization with Support Vector Machine to represent Text in Input Space (2002)
[6] Dan Hendrycks, Collin Burns, Anya Chen, Spencer Ball: CUAD: An Expert Annotated NLP Dataset for Legal Contract Review (2021)
[7] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean: A Efficient Estimation of Word Representations in Vector Space (2013)
[8] Tom M Mitchell,"Machine Learning", McGraw Hill Education, ISBN 0070428077 (1997)