HisT: A Tool for Online Hate Speech Detection

Swati Maste
Dept. of Computer Science Engineering
PES University
Bengaluru, Karnataka 560100,India

Pallavi Prabhu
Dept. of Computer Science Engineering
PES University
Bengaluru, Karnataka 560100,India

Chinmayi Shetty
Dept. of Computer Science Engineering
PES University
Bengaluru, Karnataka 560100,India

Syed Mohammad Hussain
Dept. of Computer Science Engineering
PES University
Bengaluru, Karnataka 560100,India

Dr.Art Arya
Professor, Dept. of Computer Science Engineering
PES University
Bengaluru, Karnataka 560100,India

***Abstract*—Social Media Platforms enable users to express their views. They do so in the name of Freedom of Speech. With the increased popularity of different platforms like Instagram, Facebook, Twitter, more and more people are using these platforms. Many of them give anonymity to the users, which is an added advantage for people to express their feelings without a second thought. Giving such freedom to people is dangerous on a social platform, as it can lead to people misbehaving and spreading unnecessary hate. There are some restrictions on the type of content that is acceptable. Each platform has its own rules and restrictions. But detecting the hate speech is a challenge. Hate Speech can target an individual, a particular group of people, women, people with different races, etc. It is very challenging to classify hate speech into these labels. Therefore, Hate Speech Detection and classification is a very crucial task for the moderators of these platforms, so that appropriate actions are taken against the offenders and make the internet a safe place for all humans. Therefore, this research project focuses on implementing a model for hate speech detection and classification by considering all the vital features which make a text become hate speech.**

***Index Terms*—HisT- Hate Speech Identification Smart Tool, Natural Language Processing, Transformers, Feature Engineering, Random Forest (RF), Bidirectional Encoder Representations from Transformers (BERT), Machine Learning(ML), Deep Learning(DL), Long short-term Memory**

I. Introduction

Hate Speech can be defined as "Any Tweet/comment which is specifically targeted in a way that it would hurt the sentiments of the targeted individual/Community and would bring a sense of unrest among people and disturb them on a personal level". Social Media can be a medium for users to express their perspectives [1]. With the increased usage of social media networks [2] there is no standard way for hate speech propagation. It could start with a difference of opinions between two users, or it could be simply a targeted comment in the view of freedom of expression.

Hate Speech can target an individual, or a specific group, based on race, gender, ethnicity, political views etc. [3] putting society in unrest. Therefore, it is primal to detect and classify hate speech into different subclasses/ labels. For doing so, the primary requirement is suitable feature extraction techniques. Hate Speech Detection (HSD) helps the moderators of these apps to take appropriate actions and stop the hate speech from spreading before it becomes viral and hurts the sentiment of more people. [4] The Multi-Class classification of HS, helps to understand what kind of hate speech the comment is.

Most of the existing research on hate speech detection does not focus on multistep classification of hate speech. Detecting hate speech is not simple as a binary classification [5]. Even if some of the research does focus on multistep classification, the feature extraction process of such research is not adequate, because the majority of such papers do not focus on the semantic and the emotional aspect of a text.

Another major problem in research is the presence of sarcasm in text [6]. Sarcasm by itself is a challenge to solve [7] It is furthermore difficult to classify a hate speech if there is sarcasm involved in it. Sarcasm is used when insulting or taunting a targeted person (celebrity or politician). Such tweets need to be taken care of by understanding the context. So currently, there is no research which establishes this relation. We try to address this research gap in our research work.

Hate Speech has a very negative impact on the health and morale of the online platforms. In order to prevent the propagation of hate, it is very important to detect it as early as possible. Detecting the HS and classifying it to the right subclass/labels help the moderators/authorities behind these apps to make appropriate decisions.

II. Previous Work

In [8], argue and others proposed an ensemble method through Sentic Computing to detect hate Speech. The paper tried to answer two research questions: Can Sentic Aware tools help improve performance of NLP Applications and Can a criterion be set for choosing amongst the different combinations of models or approaches proposed in order to evaluate the model performance. Authors used Affective Space, SenticNet, Similarity Based Sentiment Features(SIMON) and TF-IDF for feature extraction. Sentic Computing was done to consider the emotional aspect of the text. It is a cross-domain project which helps models understand the sentiments the way humans understand them. Datasets were collected from [9] [10] [11]

[12] The input was processed with different features, combination of these features were fed to the model by vector concatenation. SVM model was used for classification.

Watanabe et.al implemented both binary and a ternary classification [6] (Hateful, Offensive and Clean) on the combined dataset from CrowFlower platform [13] [14] and Github. The highlight of the paper was the use of four different feature extraction techniques, sentiment-based features, semantic features, unigram features and pattern features. Sentiment features were used to extract the tweet's polarity. This was extracted using a tool called "SenticStrengthScore", through which the polarity of the text is calculated. Semantic characteristics may be used to identify the stressed expressions. Any explicit type of hate speech can be identified using unigram features whereas patterns can be used to identify any longer or implicit forms of hate speech. These features were fed into different ML Models like SVM, Random Forest and a model based on ID3 called j48graft achieved the highest accuracy for binary classification with 87.4 per cent and 78.4 for ternary classification. The authors stressed about building a richer dictionary of hate speech patterns, along with a unigram dictionary to detect hateful and offensive online texts.

In [15], authors did a comparative study on different SoTA models on OLID [16] and HAS0C 2021 [17] datasets. They worked on CNN, Bi-Directional LSTM, Text-to-Text Transfer Transformer(T5) and RoBERTa. The input data was very noisy, so a number of preprocessing steps were done, like Omitting the URLs, all the texts converted to lowercase, omitting IP addresses etc. They tried out two different data augmentation methods, the first one being a token-level omission of beginning and ending tokens of every sample, and second one is autoregressive text creation using the model checkpoint by [18]. The authors observed that transformer-based architectures performed better than the Bi-LSTM and CNN models. T5+Augmented dataset displayed better results with HAS0C as opposed to the plain T5 model. On investigating the errors by the T5 model (in error analysis) on HAS0C 2021 test set, they saw that the model made 33.3 percent of misclassification on (not offensive class) and 7.520 percent of error on (hate or offensive class). This is one of the adverse effects of having an unbalanced dataset which they plan to tackle in their future work.

Du et al. proposed a sentimental context and Personal Expression Habits-Based Effective Approach for sarcasm detection. [19] Sarcasm detection is primal when it comes to the detection of hate speech intensity in a comment/ post on social media. Authors used Twitter data from Ptacek et al. [20] and Bamman and Smith [21] which contains hashtags related to sarcasm. The Reddit data used is from SARC [22]. Three subsets of this data are chosen which are politics, movies and technology. Authors used CNN to get emotive features and associated local features in the text, Semantic features were extracted using GloVE. Simultaneously, they used Bidirectional-LSTM with attention to obtain user expression habit vectors.All these three features are later combined and fed to a neural network which classifies the comment as sarcastic or non-sarcastic. The F1-score was used as the assessment metric.The F1-score obtained using this methodology was 0.76 for the Twitter dataset and 0.72 for the Reddit dataset.

In [23], Singh and others, did binary classification task to distinguish offensive tweets from non-offensive ones,the papers mainly focused on Identifying Offensive Content in Social Media Posts and do a binary classification and not on the further classification of the hate speech which we are working on.They had used BERT and DistilBERT with OLID and OLID + Offensive Language dataset, and there results were better than those produced by traditional machine learning models.They have employed a plethora of methods, from traditional machine learning to state of the transformer language models.Their results with the addition of the Offensive language dataset did not lead to a significant improvement in their performance and could not replicate the results from the OffensEval 2019 competition, which used the same Offensive language dataset.

III. Methodology

We propose HisT: Hate Speech Identification Smart Tool, that is trained on a strong feature set for the classification task, which when given a tweet classifies it as "hate" or "non hate". HisT will help in early detection of hate speech on various social media help, and thus, act vital in tackling the issue of widespread hatred.

A. Dataset

Hate Speech and Offensive Language, introduced by Davidson, is a dataset for detecting hate speech. The writers started with a hate speech dictionary developed by Hatebase.org, which included terms and phrases designated as hate speech by internet users. They found 33,458 tweets from Twitter users using the Twitter API that contained phrases from the dictionary. They pulled each user's timeline, yielding a total of 85.4 million tweets. They chose a randomized set of samples consisting of 25000 tweets containing lexical tokens and had CrowdFlower (CF) employees manually encode them from this collection. Each tweet was categorized into one of three categories: "hate", "offensive but not hate", or "neither offensive nor hate" by the workers.

B. Preprocessing

Exhaustive preprocessing was done, before sending the tweets for feature Engineering and feature selection. Various steps were undertaken as part of the preprocessing. The preprocessing was carried out in a particular order. Firstly, usernames were removed, after which URLs, special characters, white spaces and numerics were removed. After all these steps, tokenization and lemmatization were carried out. Tokenization, is a process of breaking down the text into word segments and Lemmatization was done where words were converted into their base lemma. We use SMOTE or Synthetic Minority Oversampling Technique to balance the dataset. SMOTE generates the synthetic samples for the minority class, and this helps us achieve a balanced dataset such that each target class

have an equal number of training records. This is crucial as it helps to avoid bias during model training. Using SMOTE, we successfully balanced all the datasets used for training. Addressing the skewness of training dataset highly impacts the model performance, as seen in the results during this research

C. Feature Engineering

Feature engineering is one of the most crucial part in our model, which is described before carrying forward with the experiment involving traditional machine learning models. We made use of four types of features namely TF-IDF, sentiment based, pattern features and also sarcasm based features.

1) TF-IDF: These constitute of the tweets of Davidson dataset[] undergoing pre-processing as mentioned in the previous section. These cleaned tweets where then stemmed together to form a corpus from which the TF-IDF features were extracted.

2) Sentiment-Based Features: With the help of Vader[], we were able to extract sentiment scores for each tweet[]. Based on these scores, we classified each tweet as having a 'positive', 'negative' or 'neutral' sentiment.

3) Pattern features:

D. Analysis of Models used

We experimented with the following machine learning (Random Forest) and deep learning models(LSTM, BERT), one of the key reasons to proceed with these models is their impressive performance on various tasks and flexibility of hyperparameter tuning. It led us to carry out the research to understand which models and vectorizers' are the best suitable for our problem statement. This section talks in detail about each model and their respective hyperparameters used during this experiment.

1) Random Forest: Random Decision Forests [26] is one of the ensemble learning algorithm and is a supervised learning model. One of the most peculiar characteristics of this algorithm is that it works well with both categorical and real values. It implicitly makes use of the bagging method to group the decision trees. It is an ensemble learner, which is a learning technique of enhancing the model's performance by combining numerous classifiers to solve a complicated issue. Random Forest is diverse in nature since it doesn't account for all the attributes and features while building each tree. Every tree has of its own data and features and thus makes complete usage of the CPU while building random forests as whole. Also, there is no need of splitting the data into training and testing sets, since there is always 30% of data that is unseen by the decision tree.

The features extracted (Sentiment feature, pattern feature, Sarcasm feature and TF-IDF feature) during our experiment were vectorized, combined and fed to the Random Forest model for training. Hyperparameters contribute majorly in model's learning process and with the use of optimal hyperparameters the model achieves the best performance. Hyperparameters used in the model:

- n_esimators: Number of trees the algorithm builds before averaging. The value of n_estimators in our experiment is 100.
- criterion: It is the function to measure the quality of a split, we used 'gini index' as criteria to measure the impurity.

2) Bidirectional Encoder Representations from Transformers (BERT): BERT [24] also known as Bidirectional Encoder representations from transformers is a machine learning framework that is currently being used for Natural Language Processing. This framework is based on Transformers, which is a Deep learning model where the weights are calculated based on the connection of every output element to every other input element. BERTs ability to read text input from either left-to-right and right-to-left makes it different from the others and thus, this capability enabled the introduction of transformers, known as bidirectionality. BERT Doesn't require any text pre-processing like TF-IDF vectorization, as it pre-trained on text inputs. Under the Bidirectional capability, BERT is pre-trained on two NLP tasks :

i) Masked Language Modelling.

ii) Next Sentence Prediction.

The transformer improves BERT's ability to comprehend context and ambiguity in the provided language. The transformer allows the BERT model to grasp the complete context of the term at once by examining the surrounding words, allowing it to better understand the searcher's purpose.

In our experiment, we tweaked the dense layers of BERT, making it a customized BERT architecture. We also experimented by tweaking the values of various hyperparameters of the model. The value of hyperparameters used during our

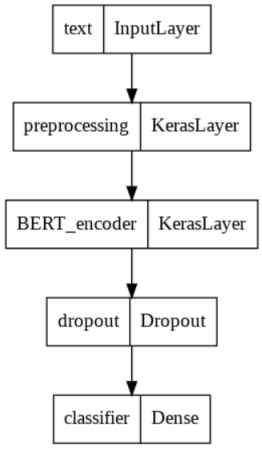

Fig. 1. Proposed Architecture of BERT Model

experiment are as listed below: The hyperparameter of dropout layer present in BERT:

- dropout_rate: The value of dropout_rate in our experiment is 0.1.

The hyperparameters of dense layer of BERT are:

- activation : None
- number of dense layers: 1
- name : classifier

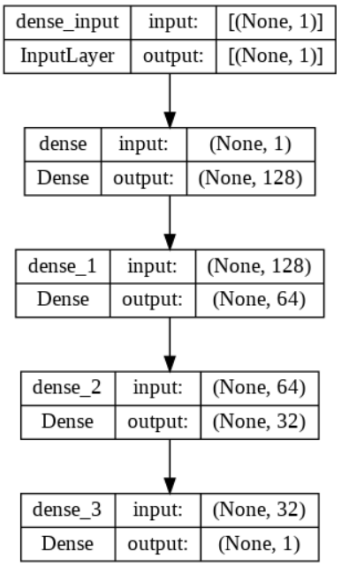

Fig. 2. The Dense layer of the BERT

Other hyperparameters used are:

- epochs : We trained our model over 5 training epochs
- optimizer : Adam- to adjust the learning rate during training
- init_lr : The initial learning rate used is 3e-5
- metric : This is the model evaluation metric, we used BinaryAccuracy.
- loss : The loss function used by us is BinaryCrossEntropy

Hyperparameter introduced by us to handle the classification task:

- threshold : We set the threshold value as 0.5 to classify each record into the respective target class.

3) Long short-term Memory(LSTM): Long short-term Memory was introduced in the paper [25], which has an improvised recurrent architecture along with suitable gradient-based learning technique. It was proposed to tackle the error backflow issues. They are popular as they are skilled in learning long-term dependencies. Every model needs features for better training, and hence, before training our model, we used GloVe (Global Vectors)[], which is a model used for extracting the vector representations of a given text. In our experiment, the training data consisting of tweets was first given to the GloVe model. The output of GloVe is the word embeddings, which was then used in the embedding layer of LSTM model and was trained for the classification task. The following figure shows the proposed LSTM architecture.

Hyperparameters we tweaked to achieve best training results, the values used in our experiment as listed below:

- epoch : We trained our model for 15 training epochs.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 100)]	0
embedding (Embedding)	(None, 100, 50)	1552700
lstm (LSTM)	(None, 100, 128)	91648
dropout (Dropout)	(None, 100, 128)	0
lstm_1 (LSTM)	(None, 100, 128)	131584
dropout_1 (Dropout)	(None, 100, 128)	0
lstm_2 (LSTM)	(None, 128)	131584
dense (Dense)	(None, 1)	129

Total params: 1,907,645
Trainable params: 354,945
Non-trainable params: 1,552,700

Fig. 3. Proposed Architecture of LSTM Model

- dropout_rate : We set the dropout rate of 0.6 in each of the dropout layers present in the proposed LSTM architecture.
- number of dense layers: Dense layers used in LSTM architecture are 1.
- activation : Activation function used in the dense layer is ”sigmoid” function.
- optimizer : Optimizer used to adjust the learning rate during the model training was ”Adam” optimizer.
- init_lr : The initial learning rate for Adam was 0.0001.
- loss: The loss function used during training was ”binary_crossentropy”

E. Metrics used

1) Accuracy: Accuracy[] is one of the evaluation metric that usually describes about the models' performance over all the target classes. This metric can be more useful whenever all the target classes have the same gravity. It can be computed as the ratio of the count of unerring predictions to the combined count of predictions. And is given as follows:

$$Accuracy = \frac{True_{positive}+True_{negative}}{True_{positive}+True_{negative}+False_{positive}+False_{negative}}$$

2) Precision: The precision[] is computed as the ratio of the count of Positive specimens unerring classifications to the combined count of specimens labelled as Positive (either correctly or incorrectly). The precision computes the model's accuracy in classifying a test record as positive. If the precision is large, we can trust the model whenever it predicts a test as Positive.

$$Precision = \frac{True_{positive}}{True_{positive}+False_{positive}}$$

3) Recall: Recall[] is computed as the ratio of the count of Positive tests precisely allotted as Positive to the combined count of Positive specimens. The recall quantifies the model's ability to ascertain Positive specimens. The higher the recall, the more positive tests are detected. This metric focuses only

on how the positive records are classified and is independent of how the negative specimens are classified.

$$Recall = \frac{True_{positive}}{True_{positive}+False_{negative}}$$

4) F-measure: F-Measure[] gives us a single score that handles both the problems of recall and precision by providing a single number. Individually, neither recall nor precision gives us the overall story of the model performance. Hence, F-measure is better as it combines the properties of both recall and precision. And it can be computed as follows:

$$F\ measure = \frac{2*Recall*Precision}{Recall+Precision}$$

5) AUC: The AUC score[] is short for Area Under the Curve. It is computed using Simpson's Rule. The higher the AUC score, the better is our classifier.

$$AUC\ Score = \int TPR\ d(FPR)$$

where, TPR is short for true positive rate and FPR is an abbreviation for false positive rate.

IV. Results

The intelligent models used help us in the classification task. In our experiment, we performed a binary classification of a tweet as "hate" or "non-hate" speech. We noticed that the dataset used, was imbalanced, and this skewness hinders the model learning, as the model can be biased towards a particular class. To tackle this issue, we oversampled using SMOTE (synthetic minority oversampling technique) to make the ratio of all the target classes equal for training.
We used Random Forest, BERT and LSTM for the classification task. The accuracy of each model is listed in figure below.

Models	Accuracy
Random Forest	94.10
BERT	82.73
GloVe+LSTM	92.75

Fig. 4. Results

We combined extensive feature extraction and Random forest model, which achieved an accuracy of 94.1% with a precision of 46.75% and an ROC-AUC score of 85.21%. BERT model was used with custom layers and achieved an accuracy of 82.73% and a F1-score of 90.5%. LSTM model with GloVe attained an accuracy of 92.75%.

V. Conclusion and Future Scope

The key takeaway of our research is the importance of feature engineering and balanced dataset. Inspired by the feature extraction done in prevailing work, we went ahead with an extensive feature engineering process, realized the importance of including both syntactic and semantic features which helped us achieve best model performance. Our work proposes the use of novel feature set, which combines 4 features, which are, Sentiment features, Sarcasm features, pattern features and the TF-IDF features. Together they form a strong feature set, as they include both syntactic and semantic features of input. These features when fed to the Random Forest model achieved the highest accuracy of 94.1%, which surpasses the Machine learning models used in the existing work[]. We were able to surpass the score of existing machine model because of two main reasons:

- SMOTE : In the existing work, the models were trained on a skewed dataset, meaning the results achieved may be biased. We overcame this training bias, by oversampling the minority class.
- Feature set : Understanding the prevailing work revealed that, most of them used different features (syntactic or semantic) with various ML models, but haven't combined them all. We propose the combination of 4 important features that can be extracted from the input, which significantly improves the model's performance, as seen in results figure above.

The future scope of our project is to find a large and balanced dataset and further perform a two-step classification of the tweets, i.e., first as "hate" or "non-hate" and further classify "hate" speech into its various subclasses.

Acknowledgment

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

References

[1] Chiril, Patricia, et al. "Emotionally informed hate speech detection: a multi-target perspective." Cognitive Computation 14.1 (2022): 322-352.

[2] Baydogan, Cem, and Bilal Alatas. "Metaheuristic ant lion and moth flame optimization-based novel approach for automatic detection of hate speech in online social networks." IEEE Access 9 (2021): 110047-110062.

[3] Mozafari, Marzieh, Reza Farahbakhsh, and Noel Crespi. "Cross-Lingual Few-Shot Hate Speech and Offensive Language Detection Using Meta Learning." IEEE Access 10 (2022): 14880-14896.

[4] Salminen, Joni, et al. "Developing an online hate classifier for multiple social media platforms." Human-centric Computing and Information Sciences 10.1 (2020): 1-34.

[5] Sahnan, Dhruv, et al. "Better Prevent than React: Deep Stratified Learning to Predict Hate Intensity of Twitter Reply Chains." 2021 IEEE International Conference on Data Mining (ICDM). IEEE, 2021.

[6] Watanabe, Hajime, Mondher Bouazizi, and Tomoaki Ohtsuki. "Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection." IEEE access 6 (2018): 13825-13835.

[7] Kumar, Akshi, et al. "Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network." IEEE access 7 (2019): 23319-23328.

[8] Araque, Oscar, and Carlos A. Iglesias. "An ensemble method for radicalization and hate speech detection online empowered by sentic computing." Cognitive Computation 14.1 (2022): 48-61.

[9] Fernandez, Miriam, Moizzah Asif, and Harith Alani. "Understanding the roots of radicalisation on twitter." Proceedings of the 10th ACM conference on web science. 2018.

[10] Rowe, Matthew, and Hassan Saif. "Mining pro-ISIS radicalisation signals from social media users." tenth international AAAI conference on web and social media. 2016.

[11] Araque, Oscar, and Carlos A. Iglesias. "An approach for radicalization detection based on emotion signals and semantic similarity." IEEE Access 8 (2020): 17877-17891.
[12] Basile, Valerio, et al. "Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women on twitter." Proceedings of the 13th international workshop on semantic evaluation. 2019.
[13] Davidson, Thomas, et al. "Automated hate speech detection and the problem of offensive language." Proceedings of the international AAAI conference on web and social media. Vol. 11. No. 1. 2017.
[14] Waseem, Zeerak, and Dirk Hovy. "Hateful symbols or hateful people? predictive features for hate speech detection on twitter." Proceedings of the NAACL student research workshop. 2016.
[15] Sabry, Sana Sabah, et al. "Hat5: Hate language identification using text-to-text transfer transformer." arXiv preprint arXiv:2202.05690 (2022).https://www.overleaf.com/project/625c3d39663cd03264948ecf
[16] Mandl, Thomas, et al. "Overview of the hasoc subtrack at fire 2021: Hate speech and offensive content identification in english and indo-aryan languages." arXiv preprint arXiv:2112.09301 (2021).
[17] Adewumi, Tosin, et al. "Sm å prat: Dialogpt for natural language generation of swedish dialogue by transfer learning." arXiv preprint arXiv:2110.06273 (2021).
[18] Adewumi, Tosin P., Foteini Liwicki, and Marcus Liwicki. "Conversational systems in machine learning from the point of view of the philosophy of science—using alime chat and related studies." Philosophies 4.3 (2019): 41.
[19] Du, Yu, et al. "An effective sarcasm detection approach based on sentimental context and individual expression habits." Cognitive Computation 14.1 (2022): 78-90.
[20] Ptáček, Tomáš, Ivan Habernal, and Jun Hong. "Sarcasm detection on czech and english twitter." Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical papers. 2014.
[21] Bamman, David, and Noah Smith. "Contextualized sarcasm detection on twitter." proceedings of the international AAAI conference on web and social media. Vol. 9. No. 1. 2015.
[22] Khodak, M., N. Saunshi, and K. Vodrahalli. "A large self-annotated corpus for sarcasm. Lang Resour Eval Conf (LREC)." (2018)
[23] Singh, Ashwin, and Rudraroop Ray. "Identifying Offensive Content in Social Media Posts." International Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situation. Springer, Cham, 2021.
[24] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
[25] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.
[26] Ho, Tin Kam. "Random decision forests." Proceedings of 3rd international conference on document analysis and recognition. Vol. 1. IEEE, 1995.