Heidelberg University
Institute of Computer Science
Database Systems Research Group

Report for the lecture Text Analytics

Hate Speech Detection

`https://github.com/fidsusj/HateSpeechDetection`

Mentor: John Ziegler

Team Member: Christopher Klammt, 3588474,
Applied Computer Science
iv249@stud.uni-heidelberg.de

Team Member: Felix Hausberger, 3661293,
Applied Computer Science
eb260@stud.uni-heidelberg.de

Team Member: Nils Krehl, 3664130,
Applied Computer Science
pu268@stud.uni-heidelberg.de

Abstract

Plagiarism statement

We certify that this report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this report has not previously been submitted for assessment in any other unit, except where specific permission has been granted from all unit coordinators involved, or at any other time in this unit, and that we have not copied in part or whole or otherwise plagiarised the work of other students and/or persons.

Contents

1 Introduction **3**

2 Related Work **4**

3 Approach **5**

3.1 Definition of Hate Speech 5

3.2 Preprocessing 5

3.3 Features 5

3.4 Dataset 6

3.5 Classifiers 7

3.6 Evaluation 8

4 Experimental setup and results **8**

4.1 Data 8

4.1.1 Data preparation 9

4.1.2 Corpus building 10

4.1.3 Data analysis 10

4.2 Evaluation method 12

4.3 Experimental details 13

4.4 Results 14

4.4.1 Feature importances 14

4.4.2 Hate speech statistics 14

4.4.3 Comparison of the classifier results 16

4.4.4 Oversampled and undersampled datasets 17

5 Analysis **17**

6 Conclusion **17**

1 Introduction

One current research area in the field of text analytics is hate speech detection. Many approaches from the recent past take use of neural network architectures to deal with such classification problems. One of the most common approaches is to use uni- and bidirectional long short-term memory (LSTM) networks, a recurrent neural network architecture that can process input of arbitrary length and remembers context information [6, 18, 17]. The paper [8] states, that even a simple gated recurrent unit (GRU) architecture can perform as good as more complex units. [1] repurposes the famous bidirectional encoder representations from transformers (BERT) language model to perform classification tasks for hate speech detection. Besides that, other approaches use convolutional neural networks (CNNs) to extract typical hate speech patterns [2, 16, 10] or even deep belief network algorithms [13]. Using neural network approaches means to automatically learn representative features for the classification task. On the other hand, the papers introduced in section 2 use a different approach by solving the classification task with manually extracted features. Nevertheless, none of the papers combines the different achievements of such recent research and compares it to a baseline neural network architecture, which is what this work is dedicated to.

After a definition of the term "hate speech" in section 3 different classifiers will be trained on a holistic, hand-crafted feature set based on recent publications in the field of hate speech detection. The task includes building and preprocessing a training corpus as well as introducing and explaining the different kinds of features. How well the different classifiers perform compared to a neural network approach as a baseline and several statictical insights into typical hate speech artifacts will be presented in section 4. The results of this work in section 5 should show which features work best for which classifier and which problems can be addressed with conventional machine learning methods and which not as opposed to neural network approaches. A summary over the achievements earned will be drawn in section 6.

2 Related Work

As the goal of this work is to solve a hate speech classification problem with manually extracted features, recent work was evaluated proposing different feature sets for this task.

[19] categorizes features into four different groups. Sentiment-based features give information about the polarity of a document, which is important as many hate speech documents stand out by being mostly negativ. Sentiment features count the occurences of punctuations, capitalized words, interjections, etc. A dictionary of typical hate speech words can be obtained by extracting most common unigrams from a given corpus uilding thr unigram features. Pattern features represent the last feature group containing common syntactic patterns based on PoS tags. The apporach presented in this paper achieved an accuracy of 87.4% using combined features from these four groups. The classifiers used were SVM, Random Forest and J48graft.

[14] concludes character n-grams, word n-grams, negative sentiment-based scores and syntactic-based features as a decent feature set to train classifiers on. Watching at the results, an optimized support vector machine with character n-grams performed best with 0.894 TPR, while optimized gradient boosting performed best with word n-grams, giving a 0.867 TPR.

[7] divides features into generic text mining features and specific hate speech detection features. Typical generic text mining features are dictionaries of insults typical for heate speech, swear words, profane words verbal abuse, etc., n-grams, lexical syntactic based template features, that capture grammatical dependencies within a sentence, topic classifications with latent dirichlet allocation, or sentiment polarity scores. On the other hand specific hate speech detection features do not rely on common abstract concepts known in the field of text analytics, but come with purpose built frameworks to detect these featues. Using the Stanford lexical parser along with a context-free lexical parsing model one can identify othering language which used a lot in hate speech. Other examples of specific hate speech features are the objectivity-subjectivity relations of the language as hate speech is more related to subjective communication, focus on particular stereotypes, intersection ism of oppression or declarations of superiority of the ingroup.

Other related work like [9], [11] and [4] once more stress the importance of word and character n-grams for hate speech detection tasks. [4] even uses count indicators for hashtags, mentions, retweets and URLs and is esspecially important as a part of the dataset used in this work originated from the project behind this paper. The best performing model achieved 0.91% overall precision, 0.9% recall and a 0.9 F1-score, but the model is biased towards classifying tweets as less hateful or offensive than the human supervisors.

3 Approach

Our epistemic research interest is to clarify the question, whether classical Machine Learning methods combined with suitable features can outperform neural network based approaches.

Figure 1 visualizes our approach (on the top) and the resulting novelties achieved through our work (at the bottom). The following chapters describe the different steps in detail. First the data is preprocessed (chapter 3.1, 3.2). The next step differs between classical Machine Learning methods and neural network based approaches. For classical Machine Learning methods an explicit feature extraction is necessary. Due to the fact that our dataset unbalanced, we further create an unbalanced, oversampled and undersampled datasets (chapter 3.4). Finally the classifiers are trained, evaluated and the results are compared (chapter 3.5, 3.6).

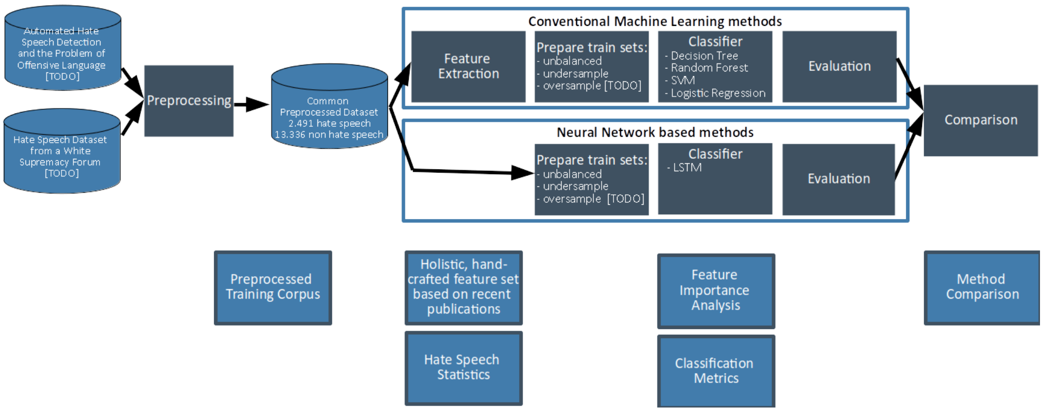

Figure 1: Approach

3.1 Definition of Hate Speech

3.2 Preprocessing

3.3 Features

For the conventional Machine Learning methods we built a hand-crafted feature set. The selection of features is based on recent works [19] and [7]. In the following the feature groups, the extraction process and the evaluation approach for the feature importances are introduced.

Feature groups

Our feature set consists of features from the following five groups:

- Unigram features [4, 7, 9, 11, 14]
  - N-grams
  - Dictionaries based on TF-IDF
- Semantic features [4, 19]
  - Number of exclamation/question/full stop marks
  - Number of capitalized words
  - Number of laughing expressions
- Pattern features [7, 14]
  - PoS-tag patterns
- Topic classification [7]
  - Latent Dirichlet Allocation
- Sentiment-based features [7, 14]
  - Polarity scores based on Vader

Feature extraction

The feature extraction of the previously mentioned features is done within a reusable pipeline, which makes it easy to add new features. Each feature is implemented as a class following a predefined structure. By adding the feature class to a list in the FeatureExtractor the feature is automatically extracted as part of the pipeline. The pipelines input are the data instances as text and the output is a dataframe containing all extracted features as numerical values.

Feature importances

3.4 Dataset

TODO: Intro

Depending on the method the inputs vary. For the classical Machine Learning methods the inputs are the extracted features as numerical values, whereas in neural network based approaches the inputs are the raw textual features. The labels are in both cases numerical values.

Then we perform the dataset balancing. TODO: Dataset balancing

Finally we receive unbalanced, undersampled and oversampled datasets.

TODO: name that oversampling currently not present for nn approaches

3.5 Classifiers

This chapter deals with the classifiers and their integration into the project.

Each classifier is trained on the training set and evaluated on the test set. So the same steps are necessary for each classifier. That is why a reusable pipeline is developed. The pipeline is developed with the Open Closed principle in mind. It is open for extensions and closed for changes. So when adding a new classifier only a few lines of code need to be adapted and steps such as finding the optimal model through hyperparameter tuning and the evaluation are done automatically as part of the pipeline.

As the methods need it, there are slight differences between the training of the classical Machine Learning methods and the neural network based approaches. Figure 2 illustrates this.

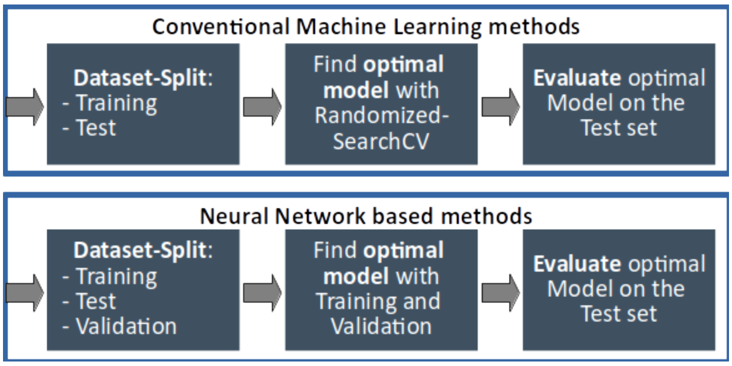

Figure 2: Classifier pipeline

For the classical Machine Learning methods the dataset is split into training (0.8) and test set (0.2). For finding the optimal model Randomized-SearchCV is executed on the defined hyperparameter search space. The hyperparameter search space was chosen based on the classifiers documentation, papers and by an empirical examination. For the decision tree [12] and for the random forest [15] were observed. Finally the model is evaluated on the test set.

TODO: Description for LR, SVM how is the hyperparameter search space set?

For neural network based approaches the dataset is again split into training (0.8) and test set (0.2). Then the training set is further split up into training (0.8) and validation (0.2). In the next step the optimal model is found by using the training and the validation set. The final step is equal to the final step in classical Machine Learning, were the model is evaluated.

To enable a performant execution we use Pythons multiprocessing library to parallelize the execution.

In this work we compare five classifiers on the different datasets (unbalanced, undersampled, oversampled). Four of them are classical Machine Learning methods (Decision Tree, Random Forest, SVM, Logistic Regression) and one neural network based approach (LSTM).

3.6 Evaluation

For evaluating the classifiers standard metrics such as accuracy, precision, recall and F1 score are used.

The accuracy specifies how many data instances are correctly classified. Only looking at the accuracy, is not that informative, because generally lots of instances are correctly classified as non hate speech, which leads to a high accuracy.

That is why also precision and recall are observed. Precision specifies how many of the predicted hate speech instances are really hate speech. A low precision means that there are lots of instances classified as hate speech, although they are not. So looking at the precision enables to detect if the trained model could built a censoring system and undermine the freedom of speech. The recall specifies how many hate speech instances are correctly classified by the model. A low recall means that there are lots of false negatives (lots of hate speech instances are not detected).

For taking into account precision and recall one can look at the F1 score.

4 Experimental setup and results

4.1 Data

There are two data sets used for the project. The first one uses data from the *Twitter API* [4][1]. It consists of a sample of around 25k tweets that were identified as hate speech based on a previously composed hate speech lexicon without regarding context information. Subsequently, each document in the corpus got labeled with one of the three categories *hate speech*, *offensive*

[1]https://github.com/t-davidson/hate-speech-and-offensive-language

language or *neutral*. Therefore the data set follows a classical ternary classification style. The workers were instructed to follow predefined definitions of each category and to take context information into consideration. Each tweet was assessed and labeled by three or more workers. The majority of tweets were classified as offensive language (76% at 2/3, 53% at 3/3), only 5% were coded as hate speech. The data is provided offline as a CSV or pickle file.

The second data set uses data from the *White Supremacy Forum* [5][2]. One document represents a sentence that is either labeled as hate or not hate. In total, 1.119 sentences containing hate and 8.537 sentences being non-hate are provided. Once again the documents were labeled manually by human actors following previously specified guidelines, on request additional context information was provided. The documents are given offline as normal text files with annotations stored in a separate CSV file.

4.1.1 Data preparation

To prepare a central dataset, both single datasets had to be transformed into a common format. For the central dataset only the class and the text content of each tweet respectively each forum contribution was considered.

The first dataset "Automated Hate Speech Detection and the Problem of Offensive Language" was entirely given as a .csv file and contains 25.297 tweets, that were either labeled as hate speech, offensive language or neither of both. To determine the right label three independent evaluators classified each tweet, the final label got assigned by the majority vote. As for the first approach, one is only interested in hate speech and neutral tweet classification, all offensive language documents in the dataset were dropped. Some tweets were retweets that were commented additionally by a user. As it could not be distinguished whether the original tweet or the retweet contains hate speech, these documents were filtered out as well. An example is shown below:

> """@jaimescudi_: ""@Tonybthrz_: ""@jaimescudi_: I swear if oomf try talking to me tomorrow.."" @"" @BarackObama"" pussy"

The original tweets can be found in between the ""..."". Same goes for tweets that cite other users without using the retweet option.

The second dataset "Hate Speech Dataset from a White Supremacy Forum" was not entirely given as a .csv file. Only the document annotations were

[2]https://github.com/Vicomtech/hate-speech-dataset

given in a .csv file, all forum contributions were stored in separate .txt files. Only documents which could not be assigned to a single class (label "idk/skip") or referred to other documents (label "relation") were dropped.

The resulting common dataset was stored in a .csv file. It contains 2.491 hate speech documents and 13.336 non hate speech documents. The dropped offensive language documents make up 17.505 instances. In case the classification results are too poor, additional 2.818 offensive language documents can be added that were labeled as hate speech by one evaluator.

4.1.2 Corpus building

The common dataset is loaded from the .csv file into a pandas dataframe. After doing basic preprocessing like removing emojis and other irrelevant characters, spacy is used to build a tokenized corpus. The language model that spacy brings decides about stop word, punctuation and white space removal. No hard coded logic or stop word lists are used in this process. This keeps URLs or other tokens including punctuation as one token. Furthermore no stemming was applied to the tokens, instead lemmatization was used as one can in this case later on use pre-trained word embeddings from i.e. Word2Vec. Furthermore tokenization works better using the lemmas instead of word stems (e.g. We'll becomes ["we","will"] and not ["we", "'ll"].

4.1.3 Data analysis

As already mentioned, the acquired dataset is an imbalanced one, which can lead to a decrease in performance and accuracy with machine learning classification. As a comparison the paper [14] also recognizes the class imbalance and tries to reduce it by applying a synthetic minority oversampling technique called SMOTE [3]. In general there are a few possibilities to tackle the challenge of unbalanced classes:

- changing the performance metric (e.g. F1-score instead of accuracy)
- undersampling, i.e. deleting instances from the over-represented class
- oversampling, i.e. adding copies of instances from the under-represented class
- generating synthetic samples (e.g by using SMOTE)

In our experiments we chose to try the different approaches and applied a simple undersampling, as well as an oversampling using SMOTE. Additionally, we used the imbalanced dataset to train a classifier to compare how this affects the performance.

In further analysis of the data, we had a look at the length of hate speech posts versus non-hate speech posts. This can be seen in Figure 3.

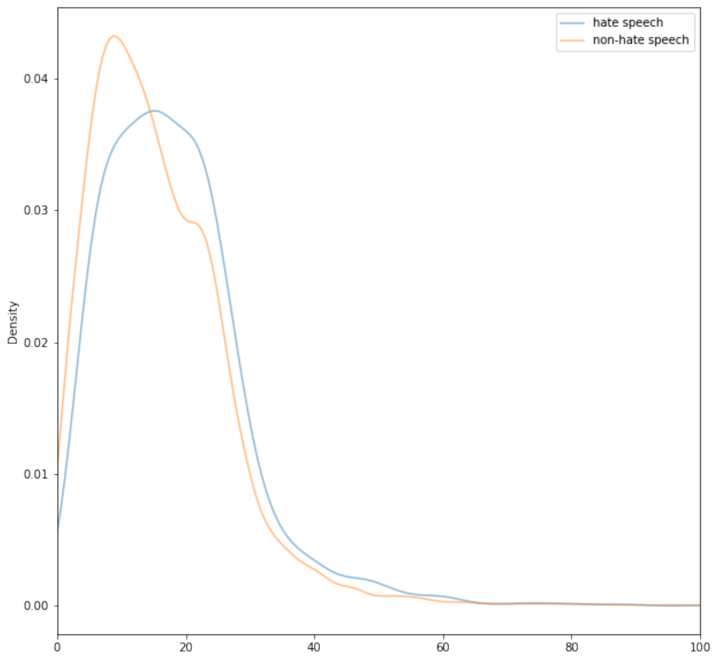

Figure 3: Density distribution of the length of a post (tweet or forum post)

Here one can see, that the hate speech posts contain more words (tokens before cleaning) than non-hate speech posts. In average a hate speech post contains 18.18 words, whereas a non-hate speech post only contains 15.85 words. Unlike expected, the hate speech posts are longer than the non-hate speech posts.

A more interesting look at the data are the most commonly used words per class. As can be seen in the word clouds in Figure 4 there are some obvious differences, such that the hate speech posts use words like “bitch”, “faggot” or “nigga”. But interestingly enough, the non-hate speech posts also often consist of the words “trash” or “white”.

(a) hate speech (b) non-hate speech

Figure 4: Word clouds

For a better insight with what data we are dealing, a few examples are shown in the following.
Examples for non-hate speech (neutral sentences):

- "billy that guy would nt leave me alone so i gave him the trudeau salute"
- "this is after a famous incident of former prime minister pierre trudeau who gave the finger to a group of protesters who were yelling antifrench sayings at him"
- "askdems arent you embarrassed that charlie rangel remains in your caucus"

Examples for hate speech:

- "california is full of white trash"
- "and yes they will steal anything from whites because they think whites owe them something so it s ok to steal"
- "why white people used to say that sex was a sin used to be a mystery to me until i saw the children of browns and mixed race children popping up all around me"

One can clearly see the hate expressed in the hate speech examples and see their discriminating nature.

4.2 Evaluation method

As part of the approach chapter 3.6 possible evaluation metrics were already introduced. Our evaluation is built upon existing quantitative metrics. Especially the F1 score is relevant, because it takes the precision (risk of censorship) and the recall (risk of ineffectiveness) into account. The f1 score is defined as follows:

$$F1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{1}$$

4.3 Experimental details

The optimal hyperparameters of the conventional Machine Learning methods are learned automatically as part of the pipeline. Nevertheless the learned optimal hyperparameters are listed in this chapter to enable the replication of our results. For readability only the hyperparameters, which differ from the default values are listed.

Optimal hyperparameters for the unbalanced dataset:

- Decision Tree: max_leaf_nodes=15, min_samples_leaf=10
- Random Forest: criterion='entropy', max_depth=10, max_features='log2'
- SVM: kernel='linear'
- Logistic Regression: C=1.2, max_iter=1500

Optimal hyperparameters for the undersampled dataset:

- Decision Tree: class_weight='balanced', criterion='entropy', max_depth=10, max_leaf_nodes=15, min_samples_split=40
- Random Forest: max_depth=10, max_features='sqrt'
- SVM: kernel='linear'
- Logistic Regression: C=1.2, max_iter=1500, solver='saga'

Optimal hyperparameters for the oversampled dataset:

- Decision Tree: class_weight='balanced', criterion='entropy'
- Random Forest: criterion='entropy', max_features='sqrt'
- SVM: all default parameters
- Logistic Regression: C=1.2, max_iter=1500

4.4 Results

For answering our research question, whether classical Machine Learning methods combined with suitable features can outperform neural network based approaches, following results were achieved:

- Investigation of feature importances (chapter 4.4.1)
- Hate speech statistics (chapter 4.4.2)
- Comparison of the classifier results (classical Machine Learning methods vs neural network based approaches) (chapter 4.4.3)
- Oversampled and undersampled datasets (chapter 4.4.4)

4.4.1 Feature importances

4.4.2 Hate speech statistics

After extracting all the features, we had a closer look at them to identify which features are characteristic for hate speech. Firstly, the semantic features in general do not signify whether a post is hate speech or not. Neither the number of exclamation marks, question marks, full stop marks, interjections or all caps words show any sign of signifying hate speech. These features are evenly distributed regarding hate speech versus non-hate speech. The only semantic features which indicate hate speech are the number of words and - to a very small degree - the number of laughing expressions. As already mentioned in subsubsection 4.1.3 the more words a post consists of, the likelier it is to be classified as hate speech (illustrated in Figure 4). Although, there are only very few laughing expressions identified per post (most do not contain any), there is a tendency for hate speech posts to contain more laughing expressions, such as “haha”, “lol” or similar.

Slightly more telling is the topic feature we trained using LDA with only 2 topics. It seems to have somewhat trained to classify into hate and non-hate - as we hoped. The hate speech posts are more likely to be classified as topic 0 than non-hate speech posts. But this difference is not really significant.

A more interesting and characteristic feature seems to be sentiment-based. As described, we extracted a sentiment-score (polarity) for each post using vader and this clearly differentiates between hate speech and non-hate speech, as shown in Figure 5. This shows, that a negative sentiment-score indicates a post being rather likely to contain hate speech. The more positive the sentiment-score is, the less probable it is classified as hate speech.

Further meaningful features were found using a dictionary approach by using the training data to generate a dictionary for hate speech and neutral

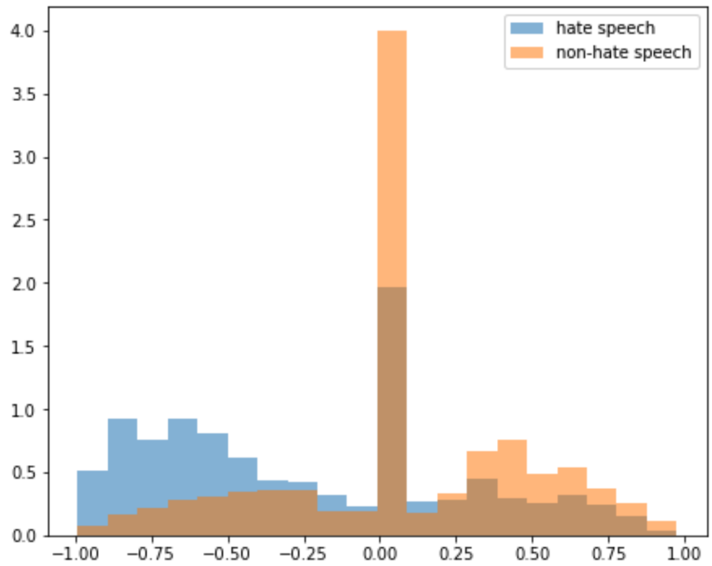

Figure 5: Normalized distribution of sentiment score for hate speech vs. non-hate speech

words. The number of hateful words is distributed such that hate speech posts contain significantly more, whereas the number of neutral words does not differ much. Examples for the most common hateful words found in hate speech posts are “fag”, “bitch”, “ass” or “nigga”. These words basically did not occur in non-hate speech posts. The most common neutral words are less informative, as the suffixes “ll” and “ve” are the most common ones for hate speech and non-hate speech posts.

Furthermore, we had an extensive look at unigrams, bigrams and trigrams for hate speech and these are significantly overrepresented in hate speech posts compared to non-hate speech posts. This especially holds true for the unigrams such as “white” which appears in 15% of hate speech posts or “not” appearing in 9% of hate speech posts. Both of these appear only half as often in non-hate speech posts. The identified bigrams only show up in a very small percentage of posts, but significantly less in non-hate speech posts. Most common bigrams for hate speech are “white trash”, “look like” and “ass nigga”.

Lastly, the feature pattern-count can somewhat indicate hate speech, as the mean amount is higher for hate speech compared to non-hate speech. As one can see in Figure 6 hate speech tends to contain more patterns (which of course were trained by using the hate speech data).

But maybe it would be worthwhile to have a closer look at the patterns

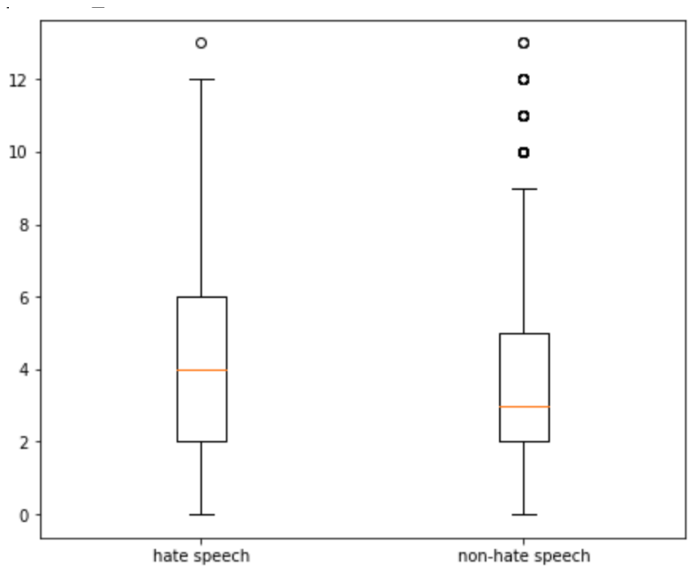

Figure 6: Boxplots comparing the number of patterns occurring in hate speech vs. non-hate speech

and adjust them for some future work. Because when we look at single patterns, some occur more often in hate speech and some more often in non-hate speech (in our dataset). For example the pattern "adjective, noun (JJ, NN)" occurs in 56% of hate speech and in 48% of non-hate speech, whereas the pattern "determiner, noun" occurs in 5% less hate speech posts than non-hate speech posts. So a more thorough analysis of these patterns could benefit this feature a lot, maybe by using individual features for each pattern. For example the biggest difference in occurrences is achieved by the pattern "personal pronoun, non-3rd person singular present verb (PRP, VBP)" with a 10% difference.

4.4.3 Comparison of the classifier results

Table 1 shows the performance metrices of the classifiers for the unbalanced dataset. All classifiers perform about equally well. And the achieved results are good with about 93% for the F1-score. So in this unbalanced case the conventional Machine Learning methods can definitely keep up with the neural network baseline.

Table 1: Classifier results for unbalanced dataset

classifier	precision	recall	accuracy	F1
Decision Tree	0.8756	0.9821	0.8671	0.9258
Random Forest	0.8809	0.9894	0.8782	0.9320
SVM	0.8697	0.9927	0.8684	0.9272
Logistic Regression	0.8831	0.9832	0.8760	0.9305
LSTM	0.9219	0.9567	0.8950	0.9390

4.4.4 Oversampled and undersampled datasets

Table 2 shows the performance metrices of the classifiers for the undersampled dataset and table 3 for the oversampled dataset.

In the undersampled case all conventional Machine Learning methods perform about equally well. Compared to the results from unbalanced dataset the conventional classifiers perform worse. In contrast the neural network baseline can keep up with the results from the unbalanced case. So in this undersampled case the conventional Machine Learning methods cannot keep up with the neural network baseline.

Table 2: Classifier results for undersampled dataset

classifier	precision	recall	accuracy	F1
Decision Tree	0.7202	0.7710	0.7431	0.7448
Random Forest	0.7261	0.8043	0.7573	0.7632
SVM	0.7193	0.8375	0.7621	0.7739
Logistic Regression	0.7246	0.8238	0.7621	0.7710
LSTM	0.9219	0.9567	0.8950	0.9390

In the oversampled case the results are located between the undersampled and the unbalanced case. An outstanding result is the Random Forest, which performs better then the other conventional Machine Learning methods. As already mentioned the SMOTE is not able to generate new textual features for generating an oversampled dataset for the neural network baseline. That is why these results are not measured.

5 Analysis

6 Conclusion

Table 3: Classifier results for oversampled dataset

classifier	precision	recall	accuracy	F1
Decision Tree	0.7973	0.7919	0.7924	0.7946
Random Forest	0.8844	0.8557	0.8701	0.8698
SVM	0.7648	0.8081	0.7767	0.7859
Logistic Regression	0.7573	0.8081	0.7713	0.7819
LSTM	not measured			

References

[1] Hind Saleh Alatawi, Areej Maatog Alhothali, and Kawthar Mustafa Moria. "Detecting White Supremacist Hate Speech Using Domain Specific Word Embedding with Deep Learning and BERT". In: *arXiv* (2020). URL: `https://arxiv.org/abs/2010.00357`.

[2] Pinkesh Badjatiya et al. "Deep Learning for Hate Speech Detection in Tweets". In: *Proceedings of the 26th International Conference on World Wide Web Companion* (2017). DOI: `10.1145/3041021.3054223`.

[3] Nitesh V. Chawla et al. "SMOTE: Synthetic Minority Over-Sampling Technique". In: *arXiv* (2011). URL: `https://arxiv.org/abs/1106.1813`.

[4] Thomas Davidson et al. "Automated Hate Speech Detection and the Problem of Offensive Language". In: *arXiv* (2017). URL: `https://arxiv.org/abs/1703.04009`.

[5] Ona De Gibert et al. "Hate Speech Dataset from a White Supremacy Forum". In: *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)* (2018). DOI: `10.18653/v1/W18-5102`.

[6] Wyatt Dorris et al. "Towards Automatic Detection and Explanation of Hate Speech and Offensive Language". In: *Proceedings of the Sixth International Workshop on Security and Privacy Analytics* (2020). DOI: `10.1145/3375708.3380312`.

[7] Paula Fortuna and SÃ©rgio Nunes. "A Survey on Automatic Detection of Hate Speech in Text". In: *ACM Computing Surveys* (2018). DOI: `10.1145/3232676`.

[8] Antigoni Maria Founta et al. "A Unified Deep Learning Architecture for Abuse Detection". In: *Proceedings of the 10th ACM Conference on Web Science* (2019). DOI: `10.1145/3292522.3326028`.

[9] Aditya Gaydhani et al. "Detecting Hate Speech and Offensive Language on Twitter Using Machine Learning: An N-Gram and TFIDF Based Approach". In: *arXiv* (2018). URL: https://arxiv.org/abs/1809.08651.

[10] Prashant Kapil, Asif Ekbal, and Dipankar Das. "Investigating Deep Learning Approaches for Hate Speech Detection in Social Media". In: *arXiv* (2020). URL: https://arxiv.org/abs/2005.14690.

[11] Shervin Malmasi and Marcos Zampieri. "Detecting Hate Speech in Social Media". In: *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)* (2017). DOI: 10.26615/978-954-452-049-6_062.

[12] Rafael Gomes Mantovani et al. *An Empirical Study on Hyperparameter Tuning of Decision Trees.* 2019. arXiv: 1812.02207 [cs.LG].

[13] Iqbal Zulfikar Muhammad, Muhammad Nasrun, and Casi Setianingsih. "Hate Speech Detection Using Global Vector and Deep Belief Network Algorithm". In: *1st International Conference on Big Data Analytics and Practices (IBDAP)* (2020). DOI: 10.1109/IBDAP50342.2020.9245467.

[14] Oluwafemi Oriola and Eduan KotzÃ©. "Evaluating Machine Learning Techniques for Detecting Offensive and Hate Speech in South African Tweets". In: *IEEE Access* 8 (2020). DOI: 10.1109/ACCESS.2020.2968173.

[15] Philipp Probst, Marvin N. Wright, and Anne-Laure Boulesteix. "Hyperparameters and Tuning Strategies for Random Forest". In: *WIREs Data Mining and Knowledge Discovery* 9.3 (2019), e1301. DOI: 10.1002/widm.1301.

[16] Pradeep Kumar Roy et al. "A Framework for Hate Speech Detection Using Deep Convolutional Neural Network". In: *IEEE Access* 8 (2020). DOI: 10.1109/ACCESS.2020.3037073.

[17] Arum Sucia Saksesi, Muhammad Nasrun, and Casi Setianingsih. "Analysis Text of Hate Speech Detection Using Recurrent Neural Network". In: *International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)* (2018). DOI: 10.1109/ICCEREC.2018.8712104.

[18] Syahrul Syafaat Syam, Budhi Irawan, and Casi Setianingsih. "Hate Speech Detection on Twitter Using Long Short-Term Memory (LSTM) Method". In: *4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)* (2019). DOI: 10.1109/ICITISEE48480.2019.9003992.

[19] Hajime Watanabe, Mondher Bouazizi, and Tomoaki Ohtsuki. “Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection”. In: *IEEE Access* 6 (2018). DOI: 10.1109/ACCESS.2018.2806394.