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

Hate speech is a very present problem, especially in the context of social media, where people can anonymously post whatever they want. While hate speech is not prohibited in the United States due to the *freedom of speech*, countries in the European Union can sue hate speech actors for offending the public order or human dignity. Automatic detection of hate speech is a very important tool to efficiently handle and moderate hate speech in social media. Based on two labeled datasets from previous papers, our work concentrated on manual feature extraction and utilizing conventional machine learning methods to perform hate speech detection. Our main research question was whether conventional machine learning methods combined with suitable features can outperform neural network based approaches. As a result we found out, that with deliberate feature extraction and optimizing machine learning methods such as Decision Tree or SVM, the results are competitive with a deep neural network approach, but they do not outperform them. Furthermore, we invested time into analyzing feature importances and hate speech characteristics, where we found out that our learned unigrams and trigrams as well as sentiment-based polarity scores indicate hate in a post. Overall, the performance we achieved with conventional machine learning methods was on par with our neural network baseline with a F1-score of 93% and an accuracy of slightly under 90%. Some further investigations were done with oversampling and undersampling the dataset, as the underlying data was unbalanced.

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 plagiarized the work of other students and/or persons.

Member contributions

Christopher Klammt

Main focus was the analysis of the preprocessed dataset and providing insights for the milestone, before extracting features. After feature extraction hate speech statistics were drawn out to show what signifies hate speech. Additionally, the semantic features and the SVM classifier were added. Before tackling the problem of the unbalanced dataset, literature research was executed to identify possible approaches. To handle the unbalanced dataset undersampling in the form of randomly deleting overrepresented instances and oversampling using synthetic generation of underrepresented instances (via SMOTE) was implemented.

Challenges were faced when trying to provide meaningful insights into the hate speech statistics as a lot of features were count-based and e.g. did not contain the occurring laughing expression or PoS pattern. Furthermore, balancing the dataset did not fare well for performance and it was not possible to apply SMOTE for the neural network approach.

Responsibility was assumed for assignment one and four.

Felix Hausberger

First proper literature research[1] was executed on which datasets to use and how to define the project idea, research question and the scope of the project including its novelties. Based on the two utilized datasets found during research the data preparation and corpus building got implemented. For the purpose of data exploration, a Word2Vec model to visualize hate speech word embeddings with t-SNE was added. Also the setup for the GitHub actions and hooks for CI was done.

Then literature research continued on which features and classifiers to use for hate speech detection[2]. Therefore, individual features were grouped and the extraction of n-gram dictionaries and a PoS-tag pattern dictionary was implemented together with the extraction of detected instance counts used as features. Furthermore, the sentiment-based polarity score using VADER and topic classification based on LDA were added as additional features. Afterwards, the Logistic Regression classifier was added to the reusable pipeline and feature importance scores were extracted for each classifier during analysis.

[1] see `https://github.com/fidsusj/HateSpeechDetection/blob/main/docs/research/papers_overview.md`

[2] see https://github.com/fidsusj/HateSpeechDetection/blob/main/docs/research/results.md

Challenges were mainly faced during the design and engineering phase of the project, i.e. which datasets, features and classifiers to use. Also the question how the data should be preprocessed and harmonized had to be dealt with. A left open challenge is to improve the PoS-tag pattern dictionary to better differentiate between hate speech and non-hate speech.

Responsibility was assumed for assignment three and four.

Nils Krehl

Literature research for the following papers [3] was done and based upon it the project idea, the research question and the scope of the project including its novelties were defined. For the purpose of data exploration, a fasttext model to visualize hate speech word embeddings with t-SNE was added. In contrast to the Word2Vec model two clusters for hate speech and neutral speech are nicely visible. One major contribution was the end to end technical architecture of the project. As part of this, reusable pipelines for feature extraction and classifier execution were developed. New features and classifiers can easily be added to the pipeline. As part of the pipeline the optimal hyperparameters are automatically learned by executing *RandomizedSearchCV* and the classifier performance metrics are calculated automatically. The Random Forest and Decision Tree classifiers as well as the neural network baseline via LSTM was realized for evaluation. Based on the literature research regarding feature groups, promising features such as TFIDF, dictionaries and special characters are implemented and visualized.

Challenges were faced in optimizing the execution time of the classifier pipeline, which made the change from *GridSearchCV* to *RandomizedSearchCV* and the use of multiprocessing necessary. Through both measures, it was possible to reduce the execution time to around 10min.

Responsibility was assumed for assignment two and four.

[3] see `https://github.com/fidsusj/HateSpeechDetection/blob/main/docs/research/papers_overview.md#ieee-vpn-required-for--marked-papers`

Contents

1 Introduction

Legal implementations on handling hate speech is different from one country to another. While hate speech is not prohibited in the United States due to the *freedom of speech*, other countries - especially in the European Union - can sue hate speech actors for either offending the public order or human dignity. While being able to prosecute actors in public without much effort, the internet and especially social media platforms provide an easy and anonymous way to practice hate speech without legal consequence enforcements. Several steps were taken to tackle hate speech online, one of them being the *code of conduct* on countering illegal hate speech online, an initiative of the european commission in close collaboration with major IT companies like *Facebook*, *Microsoft*, *Twitter* and *YouTube* [7]. While respecting the freedom of speech, these companies commit to delete hate speech contributions within 24 hours of the initial deletion request.

To further automize the process of detecting hate speech contributions, several text analytics approaches have been evaluated in the recent past. Many of them are using methods of *natural language processing* and *deep learning* for hate speech detection and rely on meaningful features being learned automatically by deep neural networks instead of using hand-crafted features. 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, 19, 18]. The paper [9] 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, 17, 11] or even deep belief network algorithms [14]. 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.

An introduction to the approach taken is given in section 3. Therefore the term "hate speech" is defined. Based upon this, the data and the features are described. Finally different classifiers will be trained and evaluated on the holistic, hand-crafted feature set based on recent publications in the field of hate speech detection. Experimental details, such as the building and

preprocessing of the training corpus as well as data insights are described in section 4. The results of this work in section 5 should show several statistical insights into typical hate speech artifacts, how the conventional classifiers performed compared to the neural network baseline and which features work best for which classifier. 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.

Watanabe, Bouazizi and Ohtsuki [20] categorize 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 negative. Sentiment features count for example the occurrences of punctuations, capitalized words and interjections. A dictionary of typical hate speech words can be obtained by extracting most common unigrams from a given corpus building the unigram features. Pattern features represent the last feature group containing common syntactic patterns based on PoS tags. The approach 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.

Oriola and Kotzé [15] propose character n-grams, word n-grams, negative sentiment-based scores and syntactic-based features as a decent feature set to train classifiers on. Looking at the results, an optimized support vector machine (SVM) 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.

Fortuna and Nunes [8] divide features into generic text mining features and specific hate speech detection features. Typical generic text mining features are for example dictionaries of insults typical for hate speech, swear words, profane words, verbal abuse, n-grams, lexical syntactic based template features, that capture grammatical dependencies within a sentence, topic classifications with latent dirichlet allocation (LDA), 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 features. Using the Stanford lexical parser along with a context-free lexical parsing model one can identify language which is 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, intersectionism of oppression or declarations of superiority of the in-group.

Other related work like [10], [12] and [4] also rely on 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 especially important as a part of the dataset used in this work originated from the project behind this paper. The best performing model achieved 91% overall precision, 90% recall and a 90% F1-score, but the model is biased towards classifying tweets as less hateful or offensive than the human supervisors.

3 Approach

3.1 Overview

The epistemic research interest in this work was to clarify the question, whether conventional machine learning methods combined with suitable features can outperform neural network based approaches.

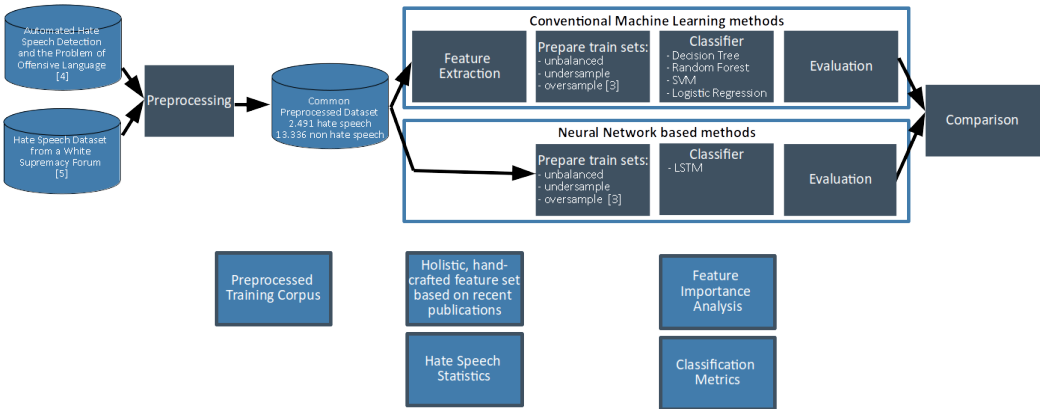

Figure 1: Approach

Figure 1 visualizes our approach (on the top) and the resulting novelties achieved through this work (at the bottom). The following chapters describe the different steps in detail. First the data was merged and preprocessed resulting in a preprocessed training corpus. The next step differs between conventional machine learning methods and neural network based approaches. For conventional machine learning methods an explicit feature extraction was necessary. Therefore achievements of recent publications were

combined to build a holistic, hand-crafted feature set. Based on these features, hate speech statistics could be further analyzed. Due to the fact that the common preprocessed corpus was unbalanced, an unbalanced, oversampled and undersampled dataset was created for further investigation. Finally the classifiers were trained, evaluated and the results were compared. Now not only the question whether conventional machine learning methods can outperform neural network based approaches could be answered, but as well which feature were most important for which classifier.

3.2 Definition of hate speech

Various definitions exist to define hate speech. This work complies with the definitions provided by the two datasets used to label the documents.

> **Ona De Gibert et al.:** Hate speech is commonly defined as any communication that disparages a target group of people based on some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic [5].

> **Thomas Davidson et al.:** A language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group [4].

One can conclude that hate speech is always targeted towards a specific group with the intention to infringe others dignity, often based on group characteristics like race, color or gender.

3.3 Data

There are two data sets used for the project. The first one uses data from the *Twitter API* [4][4]. 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 language* or *neutral*. 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% from 2/3 workers, 53%

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

from 3/3 workers), only 5% were coded as hate speech. In the end 1.430 hate speech documents, 4.175 neutral documents and 19.196 offensive language documents make up the entire dataset. The data is provided offline as a CSV or pickle file.

The second data set uses data from *Stormfront*, a white supremacy forum [5][5]. One document represents a sentence that is either labeled as hate or not hate. In total, 1.196 sentences containing hate and 9.507 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.

3.4 Features

The choice of features to use can be divided into five groups inspired by [20].

- One group of features is the unigrams features group inspired by [4, 15, 8, 10, 12]. During the training phase we extracted the top 100 words for hate speech and neutral speech based on the TFIDF scores. These words are given as dictionaries during the feature extraction phase. We therefore counted the occurrences of typical hate speech and neutral speech words in each document as a feature. A similar approach extracted typical hate speech n-grams based on word stems using NLTK during the training phase which build n-gram dictionaries. In case the number of specific n-grams surpass a certain threshold they were added to the dictionary. This threshold is 10 for unigrams, 8 for bigrams and 2 for trigrams. The number of typical hate speech unigrams, bigrams and trigrams detected was added as a feature.

- Another feature group are the semantic features taken from [4, 20]. They comprise the number of exclamation marks, question marks, full stop marks, interjections[6], all capital words, quotation marks as an approximation for the number of quotes, laughing expressions[7] and the number of words in general.

- To expand the set of features to cover syntactical characteristics of hate speech documents, PoS tag patterns were extracted belonging to the group of pattern features [15, 8]. A sliding window approach with a

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

[6] recognized by NLTKs PoS tags

[7] based on the regular expression *r"(a*ha+h[ha]*/o?l+o+l+[ol]*)|(lmao)"*

custom definable window size was used to extract PoS tag patterns for each document during the training phase. In case a hate speech pattern occurs more than 500 times in the training set it was added to a hate speech pattern dictionary. The number of hate speech patterns detected in each document was taken as a feature.

- Sentiment-based features inspired by [15, 8] were implemented by extracting the polarity scores for each document using NLTKs SentimentIntensityAnalyzer from VADER (Valence Aware Dictionary and sEntiment Reasoner). VADER is a lexicon and rule-based sentiment analysis tool specifically developed for social media sentiment analysis.

- The last feature added to the holistic, hand-crafted feature set was the topic detected from gensims latent dirichlet allocation algorithm [8]. For each document a numerical topic of either zero or one was added to the set of features based on which topic received the highest probability score.

The feature extraction of the previously mentioned features is done within a reusable pipeline, which makes it easy to add new features. Each feature was implemented as a class following a predefined interface. 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 raw text documents, stemmed documents, lemmatized documents and PoS tags each in its own dataframe column and the output is a dataframe containing all extracted features as numerical values.

3.5 Classifiers

In this work five classifiers were compared on the different datasets (unbalanced, undersampled, oversampled). Four of them are conventional machine learning methods (Decision Tree, Random Forest, SVM, Logistic Regression) and one is a neural network based approach (LSTM).

Each classifier was 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 was developed. The pipeline was 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 conventional machine learning methods and the neural network based approaches. Figure 2 illustrates this.

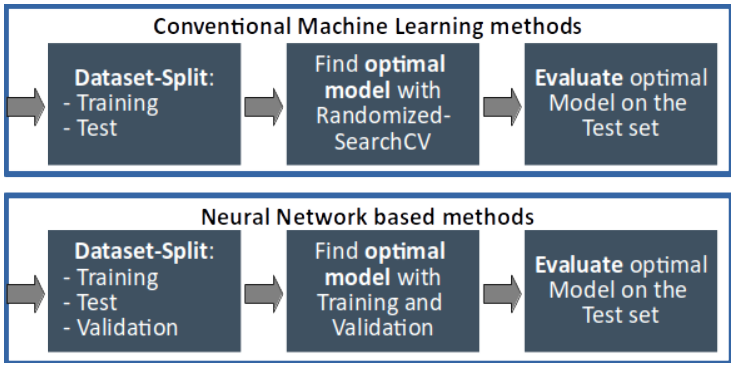

Figure 2: Classifier pipeline

For the conventional machine learning methods the dataset was split into training (0.8) and test set (0.2). In order to find the optimal model *RandomizedSearchCV* was 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 we followed [13] and for the Random Dorest [16]. For SVM we used the sklearn C-Support Vector Classification implementation and concentrated the randomized search on the kernel (linear, poly, rbf or sigmoid) as this has the biggest impact on performance. Tuning the kernel already used up quite some computational time, so we did not look into further parameters. In future work one could have a closer look at parameters such as the regularization parameter C or some kernel specific coefficients. To train the Logistic Regression model, hyperparameters were set according to the recommendations from sklearns documentation page. To try out different solver algorithms, the L2-norm penalty for regularization was chosen. As the number of samples surpass the number of features, the normal primal formulation of the regression problem was used. The search space for solver algorithms was restricted to *lbfgs*, *sag* and *saga* as others did not make sense in the training circumstances. Also different regularization strength was tested with parameter C between 0.8 and 1.2 with step width 0.1. Finally the model is evaluated on the test set.

For neural network based approaches the dataset was again split into training (0.8) and test set (0.2). Then the training set was further split up into training (0.8) and validation (0.2). In the next step the optimal model was found by using the training and the validation set. The final step was

equal to the final step in conventional machine learning, where the model is evaluated.

To enable a performant execution Pythons multiprocessing library was used to parallelize the execution.

3.6 Evaluation

For evaluating the classifiers standard metrics such as accuracy, precision, recall and F1 score were used. The accuracy specifies how many data instances are correctly classified. Only looking at the accuracy, is not that informative, because generally a lot of instances were correctly classified as non hate speech, which leads to a high accuracy. That is why also precision and recall were observed. Precision specifies how many of the predicted hate speech instances are really hate speech. A low precision means that many instances were classified as hate speech, although they are not. So looking at the precision enables us to detect if the trained model could be used as 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 were many false negatives (a lot of hate speech instances are not detected). For taking into account precision and recall one can look at the F1 score.

4 Experimental setup

4.1 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 around 25k 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. An example is shown below:

> """@DevilGrimz: @VigxRArts you're fucking gay, blacklisted hoe"" Holding out for #TehGodClan anyway http://t.co/xUCcwoetmn"

The original tweets can be found in between the ""..."". As it could not be distinguished whether the original tweet or the retweet contains hate speech, these documents were filtered out in the first version of the preprocessing pipeline. Same goes for tweets that cite other users without using the retweet option. Once the semantic feature for the number of quotes was added, the removal of quoting tweets was reverted again. Probably it is still expected behavior in social media platforms to remove contributions that build up on hate speech.

The second dataset "Hate Speech Dataset from a White Supremacy Forum" was not entirely given as a .csv file. Only the document annotations were 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 2.626 hate speech documents and 13.670 non hate speech documents. The dropped offensive language documents make up 19.196 instances.

4.2 Corpus building

The common dataset was loaded from the .csv file into a pandas dataframe. After having done basic preprocessing like lowercasing and removing emojis and other irrelevant characters, spacy was used to build a tokenized corpus. The language model from spacy decides about stop word, punctuation and white space removal. No hard coded logic or stop word lists were used in this process. This keeps URLs or other tokens including punctuation as one token.

Regarding tokenization using the lemmas instead of word stems worked better as for instance *we'll* becomes *["we","will"]* and not *["we", "'ll"]*. For getting n-gram features, the lemmatized tokens were stemmed. This keeps the range of possible unigrams for one and the same semantic word instance restricted. PoS tags were added to the dataframe using NLTKs PoS tagger as problems using spacys PoS tagger were encountered. PoS-tags are necessary for the hate speech pattern extraction.

Depending on the method the inputs vary. For the conventional machine learning methods the inputs are the extracted features as numerical values added to the dataframe by the already mentioned *FeatureExtractor* class, whereas in neural network based approaches the inputs are the raw textual contents of each document and the embeddings are learned automatically. The labels in both cases were numerical values.

Then we performed the dataset balancing. As already mentioned, the acquired dataset is an imbalanced one, which can lead to a decrease in per-

formance and accuracy with machine learning classification. As a comparison Oriola and Kotzé [15] recognized the class imbalance and tried 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 by randomly deleting instances from the over-represented class (neutral posts), as well as an oversampling using SMOTE. SMOTE does an oversampling by generating synthetic samples in the feature-space of the under-represented class, resulting in posts that statistically fall in between the different hate speech posts. Additionally, we used the imbalanced dataset to train a classifier to compare how this affects the performance.

Finally we received unbalanced, undersampled and oversampled datasets, which we used for the further experiments. In addition for all experiments, we used the F1-score instead of the accuracy as our performance metric, which should somewhat handle unbalanced classes by itself.

Unfortunately, SMOTE only works in feature space, so we could not provide an oversampled balanced dataset for our neural network approach as this works directly on the hate speech and non-hate speech posts. This could be addressed in future work.

4.3 Data insights

To better understand the dataset we are working with, the following section provides some insights into the preprocessed data corpus. In an in-depth 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.

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

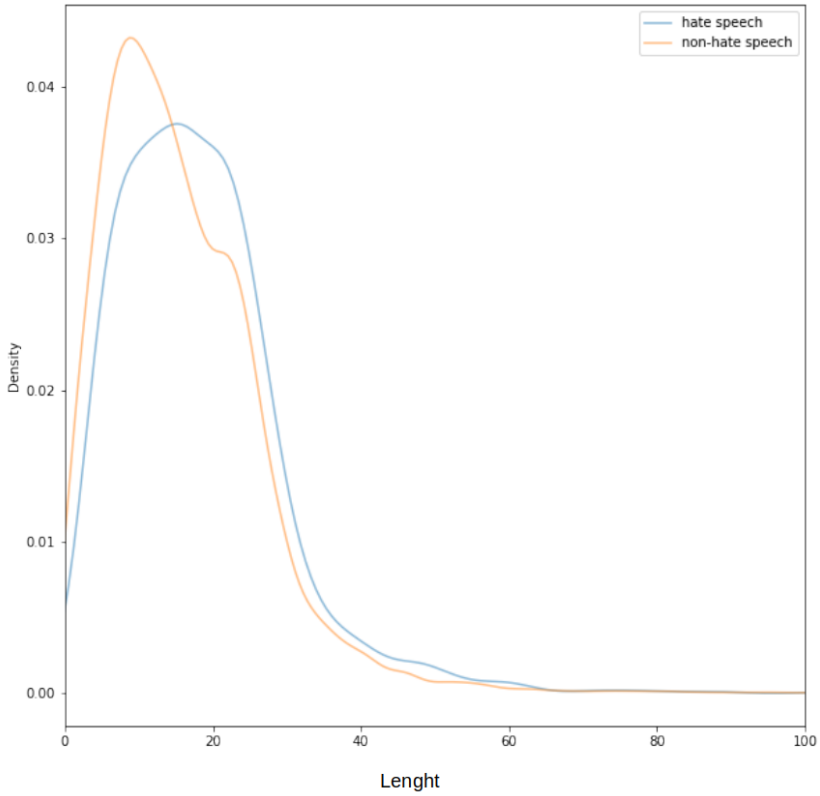

Figure 3: Density distribution of the length of a post (tweet or forum post)

words. Unlike expected, the hate speech posts are longer than the non-hate speech posts.

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

For a better insight into our data, a few examples are shown in the following.

Examples for non-hate speech (neutral sentences):

- "billy that guy wouldn’t 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"

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

Figure 4: Word clouds

- "askdems aren't 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.

4.4 Experimental details

The optimal hyperparameters of the conventional machine learning methods were learned automatically as part of the pipeline. Nevertheless the learned optimal hyperparameters are listed in the appendix A.1 to enable the replication of our results. For readability only the hyperparameters differing from the default values are listed.

5 Results and analysis

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

- Investigation of feature importances (chapter 5.1)

- Hate speech statistics (chapter 5.2)
- Comparison of the classifier results (classical machine learning methods vs neural network based approaches) (chapter 5.3)
- Oversampled and undersampled datasets (chapter 5.4)

5.1 Feature importances

Table 1: Feature importance scores per classifier

feature	**Decision Tree**	**Random Forest**	**SVM**	**Logistic Regression**
Unigrams	0.096833	0.110481	0.116233	0.340829
Bigrams	0.010989	0.030529	0.163003	0.288648
Trigrams	0.015111	0.041289	1.389380	2.009609
Hateful Words	0.165297	0.238533	0.281878	0.550001
Neutral Words	0.078015	0.060794	0.092602	0.211546
Exclamation Marks	0.015078	0.014925	0.16408	0.39866
Question Marks	0.015837	0.012568	0.077828	0.034885
Full Stop Marks	0.050472	0.039024	0.015701	0.018517
Interjections	0.001223	0.002545	0.084760	0.255209
All Caps Words	0.030842	0.025809	0.036716	0.119023
Quotation Marks	0.007124	0.011336	0.083534	0.197908
Words Total	0.176739	0.107484	0.023188	0.048423
Laughing Expressions	0.002860	0.006506	0.172888	0.150478
Pattern Count	0.094512	0.056026	0.005469	0.017724
Topic	0.030162	0.012664	0.024878	0.002830
Sentiment	0.208907	0.229479	0.447129	1.180669

The feature importance scores were extracted using the hyperparameter configuration from A.1 for the unbalanced dataset. The feature importance scores for the Decision Tree and Random Forest were calculated automatically by sklearn and equal to the gini importance score for tree classifiers. The feature importance scores for the SVM and Logistic Regression model were taken as absolute values from the feature coefficients. As the feature importances from the Decision Tree and Random Forest respectively from the SVM and Logistic Regression model are semantically not the same, the color gradient of the feature importance scores should not be compared to each other.

Probably the most important feature for almost all classifiers represents the sentiment-based polarity score. Besides the sentiment the extracted n-gram dictionaries were also informative to decide between hate speech and non-hate speech. Besides the word count any other semantic features are almost not very decisive for the task of hate speech detection and compared to n-grams and polarity scores also the hate speech pattern count and topic assignment performed weakly.

5.2 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 showed any sign of signifying hate speech. These features are evenly distributed regarding hate speech versus non-hate speech. The only semantic features which indicated hate speech are the number of words and - to a very small degree - the number of laughing expressions. As already mentioned in subsection 4.3 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 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 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

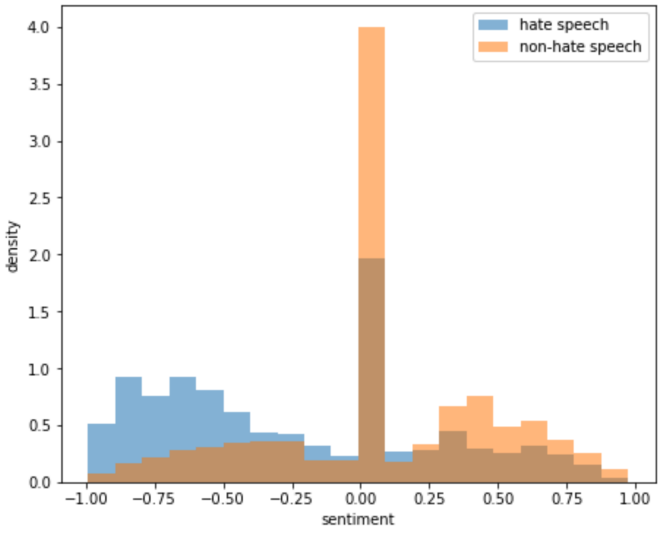

Figure 5: Normalized distribution of sentiment score for hate speech vs. non-hate speech

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 and adjust them for some future work: when we look at single patterns, some

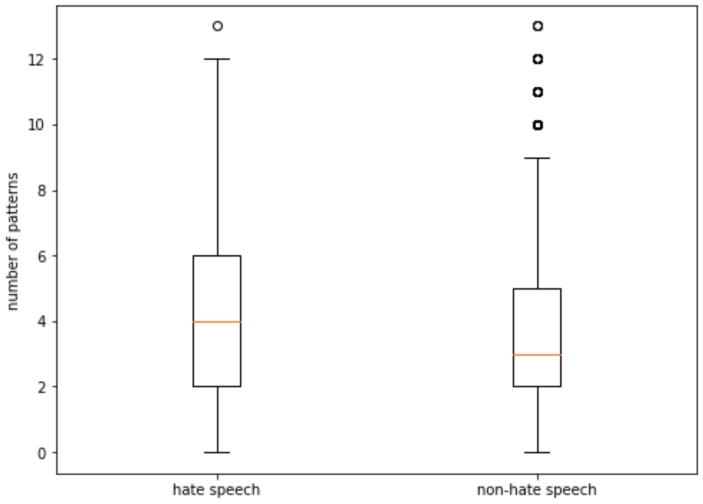

Figure 6: Boxplots comparing the number of patterns occurring in hate speech vs. non-hate speech

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.

5.3 Comparison of the classifier results

Table 2 shows the performance metrics of the classifiers for the unbalanced dataset.

Table 2: 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

All classifiers performed about equally well. And the achieved results

are good with about 93% for the F1-score. In the unbalanced case the conventional machine learning methods can definitely keep up with the neural network baseline.

5.4 Oversampled and undersampled datasets

Table 3 shows the performance metrics of the classifiers for the undersampled dataset and table 4 for the oversampled dataset.

Table 3: 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

Table 4: 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			

In the undersampled case all conventional machine learning methods performed about equally well. Compared to the results from unbalanced dataset the conventional classifiers performed worse. In contrast the neural network baseline was able to 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.

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

6 Conclusion

Coming to a conclusion, our main research question was whether conventional machine learning approaches combined with suitable features can outperform neural network based approaches. The short answer to this is, that the classical machine learning methods based on our hand-crafted feature set are definitely able to compete with our neural network baseline. For the unbalanced dataset the LSTM baseline only slightly - and rather insignificantly - outperforms the different classical methods, which all performed quite similarly. They also performed quite well with an F1-score around 93%. But it is important to note, that we used randomized search to optimize these methods by tuning the hyperparameters, so there is not much room for improvement. For our neural network approach this does not hold true, as we only used a basic implementation without much fine-tuning or testing different network architectures.

Furthermore, while answering this research question we also created a common hate speech dataset from [4] and [5] and a solid hand-crafted feature set based on recent publications upon it. We also developed an easily extendible pipeline incorporating the preprocessing of the underlying data as well as making it easy to add more features and classifiers.

Another finding of our work was the insights into the hate speech statistics and feature importance which lets us have a look at what hate speech is "made" of and indicators for it. The most important features are sentiment and unigram features, with a few words that clearly indicate hate.

As an outlook, there are some improvements and further investigations we would like to evaluate in future work. For one it should be very interesting to expand the currently binary classification into hate speech and non-hate speech to a ternary classification. As this makes the classification more complex this should be easier to achieve with neural network architectures, but it may be interesting to further evaluate the boundaries of the conventional machine learning classifiers. As already mentioned, the hyperparameter tuning for the SVM was not as extensive as it could be, so there might be room for improvement here. Additionally, the gained knowledge from the analysis of the hate speech statistics could help to further improve hate speech patterns based on PoS-tags to achieve better results and a more clear differentiation between hate speech and non-hate speech posts. One could also implement google's bad word list as a separate feature, similar to the hate speech dictionary.

A Appendix

A.1 Experimental details

Table 5: Optimal hyperparameters of the conventional machine learning methods

classifier	**unbalanced**	**undersampled**	**oversampled**
Decision Tree	max_leaf_nodes=15, min_samples_leaf=10	class_weight='balanced', criterion='entropy', max_depth=10, max_leaf_nodes=15, min_samples_split=40	class_weight='balanced', criterion='entropy'
Random Forest	criterion='entropy', max_depth=10, max_features='log2'	max_depth=10, max_features='sqrt'	criterion='entropy', max_features='sqrt'
SVM	kernel='linear'	kernel='linear'	all default parameters
Logistic Regression	C=1.2, solver='lbfgs'	C=1.2, solver='saga'	C=1.2, solver='lbfgs'

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] European Commission. "The Code of Conduct on Countering Illegal Hate Speech Online". In: (2020). URL: https://ec.europa.eu/commission/presscorner/detail/en/qanda_20_1135 (visited on 11/19/2020).

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] Oluwafemi Oriola and Eduan Kotzé. "Evaluating Machine Learning Techniques for Detecting Offensive and Hate Speech in South African Tweets". In: (2020). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8963960 (visited on 11/23/2020).

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.