Social Network Analysis and Mining (2023) 13:155
https://doi.org/10.1007/s13278-023-01156-y

ORIGINAL ARTICLE

A comparison of text preprocessing techniques for hate and offensive speech detection in Twitter

Anna Glazkova[1]

Received: 18 July 2023 / Revised: 13 October 2023 / Accepted: 14 October 2023 / Published online: 1 December 2023

Abstract
Preprocessing is a crucial step for each task related to text classification. Preprocessing can have a significant impact on classification performance, but at present there are few large-scale studies evaluating the effectiveness of preprocessing techniques and their combinations. In this work, we explore the impact of 26 widely used text preprocessing techniques on the performance of hate and offensive speech detection algorithms. We evaluate six common machine learning models, such as logistic regression, random forest, linear support vector classifier, convolutional neural network, bidirectional encoder representations from transformers (BERT), and RoBERTa, on four common Twitter benchmarks. Our results show that some preprocessing techniques are useful for improving the accuracy of models while others may even cause a loss of efficiency. In addition, the effectiveness of preprocessing techniques varies depending on the chosen dataset and the classification method. We also explore two ways to combine the techniques that have proved effective during a separate evaluation. Our results show that combining techniques can produce different results. In our experiments, combining techniques works better for traditional machine learning methods than for other methods.

Keywords Hate speech · Offensive speech · Social networks · Twitter · Preprocessing · Text classification

1 Introduction

In connection with the development of social media, harmful content gets more opportunities for spreading. Social networks and forums allow users to express their opinions freely and anonymously, which is the undoubted advantage and achievement of social media. Nevertheless, this freedom gives many opportunities to harm the psychological health of people and their mental state (MacAvaney et al. 2019). Due to the wide use of social media and the considerable number of texts contained therein, natural language processing tools play a critical role in reducing the spreading of harmful speech. Researchers indicate several types of harmful content (Mandl et al. 2019). For example, *hate speech* describes negative attributes of individuals because they are members of a particular group, *offensive speech* contains degrading and dehumanizing language, insulting an individual, threatening with violent utterances, and so on. This variety causes the challenge of implementing harmful content detection algorithms.

There is a large volume of published studies on harmful content detection (Alrehili 2019; Schmidt and Wiegand 2019; Yin and Zubiaga 2021). Most studies indicate the importance of preprocessing techniques for the performance of text preprocessing techniques for this task. Similar to other natural language processing tasks (Symeonidis et al. 2018; Kadhim 2018), ineffective text preprocessing during hate and offensive speech detection leads to confusion in machine learning algorithms.

In this work, we perform a large-scale evaluation of text preprocessing techniques on Twitter datasets for hate and offensive speech detection. To our knowledge, we present the most comprehensive investigation so far of tweet preprocessing for the task. We separately evaluate 26 common techniques, which we divide into eight types. We perform our experiments on four harmful content detection benchmarks. We use six approaches to text classification: three of them are traditional (logistic regression, random forest, and linear support vector classifier), while others are based on deep learning paradigms (convolutional neural networks, bidirectional encoder representations from transformers (BERT),

✉ Anna Glazkova
a.v.glazkova@utmn.ru

[1] School of Computer Science, University of Tyumen, 15a Perekopskaya street, Tyumen, Russia 625003

and RoBERTa). Therefore, we first carried out an extensive comparison of text preprocessing techniques for Twitter texts using two transformer-based models. Since transformers are currently widely used for this task and demonstrate state-of-the-art results on many benchmarks, these results are important for social network researchers and machine learning specialists. Our results show that some preprocessing techniques can increase the model's performance, while others decrease the scores. We also demonstrate that the efficiency of text preprocessing depends on the selected approach to text classification and the characteristics of the dataset. Thus, selecting a classification model and formatting the dataset are crucial steps for hate and offensive speech detection. We identify effective techniques separately per datasets and models. Then we experiment with two ways to combine techniques.

The main contributions of the paper can be summarized as follows: (a) numerous preprocessing techniques were evaluated and analyzed in terms of their effectiveness on several Twitter datasets and machine learning models; (b) two strategies for combining techniques were investigated; (c) it was shown that the choice of preprocessing techniques affects the classification performance; at the same time, the characteristics of the dataset are very important and they often play a key role in the effectiveness of preprocessing. The paper is organized as follows. Section 2 presents a brief exploration of related work. Section 3 contains the description of the considered text preprocessing techniques, utilized datasets, models, and evaluation metrics. Section 4 reports and discusses the results for the separate use of techniques and for technique combination. Section 5 describes the limitations of the study. Section 6 concludes this paper.

2 Related work

The impact of text preprocessing for the task of hate and offensive speech detection is widely discussed by many scholars. Previous studies investigated the impact of different preprocessing techniques and attempted to make general conclusions on their contribution to the results of text classification.

To date, several competitions in hate and offensive speech detection have been held as part of major workshops on natural language processing. The organizers of these competitions presented their overviews that generalize the results obtained by participants, including issues related to the utilized text preprocessing techniques. For instance, during the Semeval-2019 shared task related to multilingual detection of hate speech against immigrants and women in Twitter (Basile et al. 2019), most of the submitted systems adopted traditional preprocessing techniques, such as tokenization, lowercase, stopwords, URLs, and punctuation removal. Some participants investigated Twitter-driven preprocessing procedures such as splitting hashtags into separate words, converting slang into correct English, and converting emoji into words. In particular, the authors of Montejo-Ráez (2019) converted all the mentions to a common tag and tokenized hashtags. In Ameer et al. (2019), the texts were stemmed and cleaned of stopwords. The authors of Garain and Basu (2019) utilized removing links, mentions, and spaces.

During the Semeval-2020, the shared task on multilingual offensive language identification in social media was conducted (Zampieri et al. 2020). Most teams performed some kind of preprocessing or text normalization. The most common preprocessing techniques were converting emojis to plain text, segmenting hashtags, providing the expansion of abbreviations, replacing profane words, correcting errors, lowercasing, stemming, and/or lemmatizing. Other techniques included the removal of users' mentions, URLs, hashtags, emojis, special characters, and/or stopwords. The winner of the Task A (Offensive Language Detection) (Wiedemann et al. 2020) used a RoBERTa-based model (Liu et al. 2019). The winning solution of Task B (Categorization of Offensive Language) and Task C (Offensive Language Target Identification) (Wang et al. 2020) represented a multilingual method using pretrained language models: ERNIE (Zhang et al. 2019) and XLM-R (Conneau et al. 2020). Neither participant specified any preprocessing techniques in their papers.

The shared tasks on hate and offensive speech detection for English were also conducted as a part of the HASOC competitions in 2019–2021 (Mandl et al. 2019, 2020; Modha et al. 2021). The winner of HASOC2019 (Wang et al. 2019) proposed an LSTM-based approach (Hochreiter and Schmidhuber 1997) and used the following preprocessing scheme. The words were retained for hashtags; username mentions were tokenized; all contractions were split into two tokens; and emoji were replaced with the corresponding words by emotion lexicons. In 2020, the first-place solution (Mishra et al. 2020) was based on an LSTM that used GloVe embeddings (Pennington et al. 2014) as input. Initially, the texts were converted into lowercase, then all punctuation marks were removed from the texts. In 2021, the winner of the task did not submit a system description paper. The second- and third-place (Bölücü and Canbay 2021; Glazkova et al. 2021) solutions were based on graph convolutional network (GCN) (Wang et al. 2020; Liao et al. 2021) and Twitter-RoBERTa (Barbieri et al. 2020), respectively. In the first case, the authors used tokenization of hashtags, removed repeated characters, and preprocessed emphasis and censored words. In the second case, all users' mentions were replaced with a special placeholder, and the URLs were removed.

Other recent shared tasks related to hate and offensive speech detection in English posts include shared tasks on toxic span detection (Pavlopoulos et al. 2021), identification of hate and offensive speech in code-mixed postings (Modha et al. 2022), online sexism detection (Kirk et al. 2023), and identification of hate speech in multimodal content (Thapa et al. 2023). These shared tasks are summarized in Table 1.

In addition to the listed studies, some authors performed a comparison of text preprocessing techniques for the task of hate and offensive speech detection. A large-scale study on comparison text preprocessing techniques for hate speech detection on Twitter was presented in Naseem et al. (2021). The authors compared twelve preprocessing techniques for both traditional and deep learning classifiers. Deep learning approaches included convolutional neural networks, LSTM, and BiLSTM. The authors recommended a combination of preprocessing techniques based on the experiments on three datasets. The best-performing techniques were lemmatization and lower-casing of words, while the worst-performing techniques were removing punctuation, URLs, users' mentions, and hashtag symbols. Results varied with different learning algorithms, which confirmed that choosing a suitable learning algorithm is a considerable factor in text classification performance. The authors stressed the importance of the investigation of various combinations of preprocessing techniques and their interactions. In this study, some different similar techniques were evaluated together. For example, the removal of URLs, hashtags, and mentions were performed simultaneously.

Since harmful content detection and sentiment analysis are close tasks (Zhou et al. 2021; Plaza-Del-Arco et al. 2021), we also investigated research related to evaluating the effectiveness of text preprocessing techniques for sentiment analysis of tweets. In Angiani et al. (2016), the authors compared several preprocessing techniques (stemming, removal of stopwords, processing of emoticons) utilizing the naive Bayes classifier. They achieved an improvement over the baseline result through the use of stemming. A comparison of 16 preprocessing techniques across four traditional machine learning algorithms on two datasets was presented in Symeonidis et al. (2018). Lemmatization, removing numbers, and replacing contractions improved the performance of classifiers, while others did not. The authors of Alam and Yao (2019) showed that the accuracy of some traditional machine learning algorithms can be significantly improved after applying several preprocessing steps: removing emoticons and stopwords and stemming. In Ramachandran and Parvathi (2019), the authors showed a positive effect of stopwords removal for the performance of the naive Bayes classifier. A summary of the listed studies is provided in Table 2. Research on the effectiveness of preprocessing techniques for sentiment analysis is not limited to using only Twitter texts. A number of studies investigated features for sentiment analysis of spam reviews (Saeed et al. 2018, 2020, 2021, 2022), messages from StockTwits (Renault 2020), news (Štrimaitis et al. 2021; Dogru et al. 2021; Oliveira and Merschmann 2021), etc.

Text preprocessing is an important step for creating classification models. Currently, a large number of studies on hate and offensive speech detection have been performed. In addition, several works were devoted to the comparison of text preprocessing techniques for this task. Existing research on tweet preprocessing techniques for hate and offensive speech detection mostly evaluates the effectiveness of technique combinations or limited technique types. This study is aimed at overcoming this research gap. We perform a large-scale comparison of separate preprocessing techniques on several hate and offensive speech detection benchmarks. In addition, we first conduct a large-scale analysis of two transformer-based models. We try combinations of individual effective techniques, as well as experiment with combining techniques using two ways for technique generalization.

3 Methods

3.1 Preprocessing techniques

In this study, 26 commonly used preprocessing techniques were evaluated. All considered techniques were divided into the following types:

- basic techniques, such as converting to lowercase, lemmatizing, stemming, and removing special characters;
- handling of digits (removing, tokenizing, and converting to words);
- handling of URLs (removing and tokenizing);
- handling of mentions (removing and tokenizing);
- handling of emoji and emoticons (removing, tokenizing, and converting to textual description);
- handling of hashtags (removing, tokenizing, and segmenting into separate words);
- lexical transformations, i.e., removing stopwords, replacing decontractions and acronyms, tokenizing profane lexicon;
- corrections, including spelling correction and removing repeated letters.

Sections 3.1.1 through 3.1.8 contain detailed descriptions of each considered type. For better presentation, a correspondence between preprocessing techniques utilized in this work and their sequential numbers is listed in Table 3. To implement preprocessing techniques, the following libraries were used: tweet-preprocessor,[1] NLTK (Bird 2006), num2words,[2]

[1] https://github.com/s/preprocessor.

[2] https://github.com/savoirfairelinux/num2words.

Table 1 Summary of recent shared tasks aimed at hate and offensive speech detection in English

References	Aim	Data source	# Labeled posts	Classes
Basile et al. (2019)	Detecting hate speech against immigrants and women	Twitter	10,000	(1) Hateful/non-hateful; (2) individual target/generic; (3) aggressive/non-aggressive
Mandl et al. (2019)	Detecting hate, offensive, and profane speech	Twitter, Facebook	7,005	(1) Hate and offensive/non-hate–offensive; (2) hateful/offensive/profane; (3) targeted insult/untargeted
Zampieri et al. (2019)	Detecting offensive speech	Twitter	14,100	(1) Offensive/non-offensive; (2) targeted insult/untargeted; (3) individual target/group/other
Zampieri et al. (2020)	Detecting offensive speech	Twitter	9,093,037 (Task A), 190,896 (Task B), 150,399 (Task C). The dataset was labeled in a semi-supervised manner	(1) Offensive/non-offensive; (2) targeted insult/untargeted; (3) individual target/group/other
Mandl et al. (2020)	Detecting hate, offensive, and profane speech	Twitter	4,522	(1) Hate and offensive/non-hate–offensive; (2) hateful/offensive/profane
Pavlopoulos et al. (2021)	Detecting the spans that make a post toxic when detecting such spans is possible	Civil Comments	10,629	Systems had to extract a list of toxic spans, or an empty list, per post
Modha et al. (2021)	Detecting hate, offensive, and profane speech	Twitter	5,124	(1) Hate and offensive/non-hate–offensive; (2) hateful/offensive/profane
Modha et al. (2022)	Identifying hate speech and offensive language offered in code-mixed postings in Hinglish (Hindi + English) and German	Twitter	6,298 (Task 1, Hinglish and German), 5,910 (Task 2, Hinglish and German)	(1) Hate and offensive/non-hate–offensive; (2) standalone hate/contextual hate/non-hate
Kirk et al. (2023)	Detecting and classifying online sexism	Gab, Reddit	20,000 (Task A), 4,854 (Tasks B and C)	(1) Sexist/non-sexist; (2) four categories of sexism; (3) eleven fine-grained sexism vectors
Thapa et al. (2023)	Detecting hate speech in multimodel online content (text-embedded image)	Twitter, Facebook, Reddit	4,723 (Bhandari et al. 2023)	(1) Hateful/non-hateful; (2) target detection (community, individual, or organization)

Table 2 Summary of studies performing an evaluation of preprocessing techniques for hate speech detection and sentiment analysis for Twitter datasets

References	Number of datasets	Task	Models	Techniques
Naseem et al. (2021)	3	Hate speech detection	Convolutional Neural Network, Decision Tree, Naive Bayes, Logistic Regression, Random Forest, Recurrent Neural Network, Support Vector Machines	Decontraction; lemmatization; lowercasing; removal of numbers; removal of punctuation; removal of repeated letters; removal of stopwords; removal of URLs, mentions, and hashtags; replacement of abbreviations and slang; replacement of emoticons; spelling correction; word segmentation
Angiani et al. (2016)	2	Sentiment analysis	Naive Bayes	Processing of emoticons; removal of stopwords; stemming
Symeonidis et al. (2018)	2	Sentiment analysis	Convolutional Neural Network, Logistic Regression, Naive Bayes, Support Vector Machines	Decontraction; handling capitalized words; handling negations; lemmatization; lowercasing; part of speech tagging; removal of numbers; removal of Unicode string and noise; removal of punctuation; removing repetitions of punctuation; removal of stopwords; replacement of elongated words; replacement of slang and abbreviations; replacement of negations with antonyms; replacement of URLs and mentions; spelling correction; stemming
Alam and Yao (2019)	1	Sentiment analysis	Maximum Entropy, Naive Bayes, Support Vector Machines	Removal of emoticons; removal of stopwords; stemming
Ramachandran and Parvathi (2019)	1	Sentiment analysis	Naive Bayes	Part-of-speech tagging, removal of stopwords

Table 3 Preprocessing techniques

№	Technique	Type	Example (original tweet)	Example (transformed tweet)
1	Lowercase	Basic techniques	"Bae" sounds like such a ghetto word. Use something else	"bae" sound like such a ghetto word. use something else
2	Lemmatization			"Bae" sound like such a ghetto word. Use something else
3	Stemming			"Bae" sound like such a ghetto word. Use someth els
4	Removing special characters			Bae sound like such a ghetto word Use something else
5	Removing digits	Handling of digits	7 days of work down, 5 more to go ☹☹ end help and chocolate x	days of work down, more to go ☹☹ end help and chocolate x
6	Tokenizing digits			$NUMBER$ days of work down, $NUMBER$ more to go ☹☹ end help and chocolate x
7	Converting digits to words			seven days of work down, five more to go ☹☹ end help and chocolate x
8	Removing URLs	Handling of URLs	Hope you're not one of them sir? https://t.co/bnIaHuBe1r	Hope you're not one of them sir?
9	Tokenizing URLs			Hope you're not one of them sir? URL
10	Removing mentions	Handling of mentions	#preprocessing is a cruical part of @ML projects	#preprocessing is a cruical part of projects
11	Tokenizing mentions			#preprocessing is a cruical part of $MENTION$ projects
12	Removing emoji	Handling of emoji and emoticons	@username miss u so much... :(☹☹	@username miss u so much... :(
13	Tokenizing emoji			@username miss u so much... :($EMOJI$ $EMOJI$
14	Converting emoji to words			@username miss u so much... :(:unamused_face::unamused_face:
15	Removing emoticons			@username miss u so much... ☹☹
16	Tokenizing emoticons			@username miss u so much... $SMILEY$ ☹☹
17	Converting emoticons to words			@username miss u so much... sad ☹☹
18	Removing hashtags	Handling of hashtags	i get to see my daddy today!! #80days #gettingfed	i get to see my daddy today!!
19	Tokenizing hashtags			i get to see my daddy today!! $HASHTAG$ $HASHTAG$
20	Hashtag segmentation			i get to see my daddy today!! 80 days getting fed
21	Removing stopwords	Lexical transformations	@BoneSaw00 There's a reason why the Yankees don't want Anderson any longer	@BoneSaw00 There's reason Yankees want Anderson longer.
22	Decontraction		You're my person ☺	You are my person ☺
23	Replacing acronyms		We don't need this. Lol	We don't need this. laughing out loud
24	Tokenizing profanity		RT @incorrectjeon: I fucking love this friendship	RT @incorrectjeon: I profanity love this friendship
25	Spelling correction	Corrections	THEYRE CAGTING BTS #BBMA	THEYRE CASTING ITS #BBMA
26	Removing repetitions		he fine , he sooooooo fine	he fine , he so fine

emoji,[3] pyspellchecker,[4] and better_profanity.[5] Stemming was performed using the Snowball stemmer (Porter 2001).

3.1.1 Basic techniques

The most common techniques used for preprocessing all types of texts are translation in the lowercase (1), normalization (lemmatization (2) or stemming (3)), and removing special characters (4). These techniques were used to train baseline models. In Sect.3.1, we experiment with a consistent exclusion of basic techniques from the baseline.

3.1.2 Handling of digits

Common approaches to preprocessing digits in the text are removing digits (5), tokenizing digits (6), and their conversion to words (7). Tokenization means replacing digits with special tokens. For example, consider the following original tweet:

[3] https://github.com/carpedm20/emoji.

[4] https://github.com/barrust/pyspellchecker.

[5] https://github.com/snguyenthanh/better_profanity.

7 days of work down, 5 more to go ☹☹ end help and chocolate x

The transformed version after removing digits is:

days of work down, more to go ☹☹ end help and chocolate x

After tokenizing:

$NUMBER$ days of work down, $NUMBER$ more to go ☹☹ end help and chocolate x

After converting digits to words:

seven days of work down, five more to go ☹☹ end help and chocolate x

Some previous studies Nobata et al. (2016); Luu et al. (2020) used the preprocessing of digits as a step of data preparation for hate speech detection.

3.1.3 Handling of URLs

We explored two popular techniques for preprocessing URLs, such as removing URLs (8) and tokenizing URLs (9). For instance, the following tweet:

Hope you're not one of them sir? https://t.co/bnIaHuBe1r

will be transformed after removing URLs to:

Hope you're not one of them sir?

and after tokenizing URLs to:

Hope you're not one of them sir? $URL$

Preprocessing of URLs is one of the most common steps for tweet preparation (Banerjee et al. 2021; Menini et al. 2021).

3.1.4 Handling of mentions

For user mentions, we used the same techniques as for the previous class, i.e., removing mentions (10) and tokenizing mentions (11). A tweet contains mentions when it includes another person's username anywhere in its text. User mentions start with the "@" symbol. For example, the tweet:

#preprocessing is a crucial part of @ML projects.

will be transformed after removing mentions to:

#preprocessing is a crucial part of projects.

and after tokenizing mentions to:

#preprocessing is a crucial part of $MENTION$ projects.

Researchers often use preprocessing of mentions for tweet analysis, e.g., in Glazkova et al. (2021) and Banerjee et al. (2021).

3.1.5 Handling of emoji and emoticons

Emoticons represent ordinary punctuation marks from a standard computer keyboard to build up a representation of a face with a particular expression, while emoji are graphic symbols with predefined names and codes (Bai et al. 2019). Here we evaluated the following techniques: removing emoji (12), tokenizing emoji (13), converting emoji to textual description (14), removing emoticons (15), tokenizing emoticons (16), and converting emoticons to textual description (17). The list of emoticons is given in "Appendix A."

For example, consider the following tweet:

@username miss u so much... :(☹☹

It will be transformed the following way:

@username miss u so much... :((removing emoji);

@username miss u so much... :($EMOJI$ $EMOJI$ (tokenizing emoji);

@username miss u so much... : (:unamused_face::unamused_face: (converting emoji to words);

@username miss u so much... ☹☹(removing emoticons);

@username miss u so much... $SMILEY$ ☹☹(tokenizing emoticons);

@username miss u so much... sad ☹☹ (converting emoticons to words).

Many researchers, in particular, the authors of Alshalan and Al-Khalifa (2020) and Ranasinghe and Hettiarachchi (2020), utilized preprocessing emoji and emoticons during hate and offensive speech detection.

3.1.6 Handling of hashtags

We considered the following techniques for preprocessing hashtags: removing hashtags (18), tokenizing hashtags (19), and hashtag segmentation (20). Hashtag segmentation is the technique that breaks a hashtag into its constituent tokens and removes the "#" symbol (Kodali et al. 2022). To split hashtags, we utilized the implementation of the maximum matching algorithm,[6] i.e., given a string s, get all possible segmentations of s into dictionary words, then return the "longest" segmentation (Reuter et al. 2016). This implementation utilizes an English vocabulary from NLTK.

[6] https://github.com/matchado/HashTagSplitter.

For example, the tweet:

i get to see my daddy today!! #80days #gettingfed

will be transformed to:

i get to see my daddy today!! (removing hashtags);

i get to see my daddy today!! $HASHTAG$ $HASHTAG$ (tokenizing hashtags);

i get to see my daddy today!! 80 days getting fed (hashtag segmentation).

Previous studies widely used hashtag preprocessing for hate and offensive speech detection, for example in (Caselli et al. 2020; Toraman et al. 2022).

3.1.7 Lexical transformations

Here we considered several techniques related to the preprocessing of vocabulary. The first technique in the class is removing stopwords (21). Stopwords are extremely common words that are often excluded from texts because of their high prevalence, such as *the*, *on*, *that*, and *of*. This technique is widely used for tweet preprocessing, for example in Alshalan and Al-Khalifa (2020) and Das et al. (2021). The second technique in this class is decontraction (22), which replaces shortened combinations of functional words with their full analogues, e.g., *I'll* → *I will*. In particular, decontraction was used in Naseem et al. (2019) for hate speech detection. The next technique is similar to the previous one, but we replace common online acronyms (23) instead of functional words, for example, *gr8* → *great*. The lists of contractions and acronyms are given in "Appendix A." The last technique in this class is tokenizing profane lexicon (24), i.e., replacing profane words with a token *profanity* using the better_profanity package.

3.1.8 Corrections

The last class contains two techniques related to correcting and normalizing spelling. The first technique is spelling correction (25) using a Levenshtein distance algorithm and the pyspecllchecker package. The second technique is correcting words with repeated letters (26), for instance, *yeeeeees* → *yes*. These techniques are commonly utilized in tweet analysis, for example in Mohammad (2018) and Hu et al. (2020).

3.2 Datasets

To date, several datasets for hate and offensive speech detection on Twitter have been published. Most of them consist of tweets manually labeled as hate/offensive and neutral. In this work, the task of binary classification was considered since such task formulation is the most common (Fortuna and Nunes 2018; Poletto et al. 2021). The following datasets were utilized:

- HateBase (Davidson et al. 2017), the dataset contains tweets of three categories: hate speech, offensive but not hate speech, or neither offensive nor hate speech (neutral). In this work, the dataset was divided into two subsets. The first comprises hate and neutral tweets (**hb_hate**). The second consists of offensive and neutral tweets (**hb_off**). This division allows us to compare the preprocessing techniques that are effective for hate and offensive speech detection, respectively.
- HASOC2020 (**hasoc**) (Mandl et al. 2020), the corpus has two labels: neutral tweets and tweets containing hate, offensive, or profane content.
- OLID (**olid**) (Zampieri et al. 2019), this dataset contains examples of two classes, namely offensive and not offensive. The first class includes tweets containing inappropriate language, insults, or threats. The texts from the second class are neither offensive nor profane.
- HatEval (**heval**) (Basile et al. 2019), the corpus was collected for detecting hateful content in social media texts, specifically in Twitter's posts, against two targets: immigrants and women. It consists of neutral and hateful tweets.

Most hate and offensive speech datasets are sampled by crawling social media platforms using keywords considered relevant for harmful content, scrapping hashtags, or extracting timelines for harmful content spreaders. All these methods may introduce a bias because this process might limit the collection to topics and words that are remembered (Mandl et al. 2020). Empirical studies (Wiegand et al. 2019; Davidson et al. 2019) have pointed out that these approaches may lead to bias. The following approaches were used to collect the datasets listed above. The authors of HateBase utilized the lexicon containing words and phrases identified by internet users as hate speech, compiled by Hatebase.org. They searched for tweets containing terms from the lexicon and extracted the timeline for each communicant. From this corpus, the authors took a random sample of tweets containing terms from the lexicon and had them manually coded by CrowdFlower workers. To obtain potentially hateful tweets for HASOC2020, the support vector classifier was trained on the OLID and HASOC2019 (Mandl et al. 2019) datasets. The authors of the dataset considered all the tweets that were classified as hateful by the week classifier and added five percent of the tweets that were not classified as hateful randomly. Then this set of English tweets was manually annotated. For OLID, the crowd-sourcing platform Figure Eight was used for annotation. For HatEval, the data have been collected using different gathering strategies: (1) monitoring potential victims of hate accounts, (2) downloading the history of identified haters, and (3) filtering Twitter streams with keywords, i.e., words, hashtags, and stems. The most part of the training set of tweets against women has been derived from an earlier collection carried out in the context of two previous challenges on misogyny identification (Fersini et al. 2018a, b).

The data statistics are summarized in Table 4. The average length is calculated in terms of the count of tokens determined using NLTK (Bird 2006). The users' mentions and URLs are represented in the OLID dataset in a unified manner. Therefore, we have not applied tokenizing URLs (technique 9) and tokenizing mentions (11) to this dataset.

3.3 Models

To provide a comprehensive comparison of preprocessing techniques, we chose five supervised and deep learning methods.

- Logistic Regression (**LR**), a supervised machine learning method that analyzes the relationship between the variables and classifies data into discrete classes. In Oriola et al. (2020), LR was used for detecting offensive and hate speech in South African tweets. Also some researchers (Ashraf et al. 2022; Huang et al. 2020; Silva et al. 2020) used LR as a baseline.
- Random Forest (**RF**), an ensemble supervised method that constructs a set of decision tree at the training time. The authors of Alfina et al. (2017) and Nugroho et al. (2019) used RF for detecting hate speech in social media.
- Linear Support Vector Classifier (**LSVC**), a support vector machine model that applies a linear kernel function to perform classification. LSVC was applied for hate and offensive speech detection in Balouchzahi and Shashirekha (2020) and Fromknecht and Palmer (2020).
- Convolutional Neural Network (**CNN**), an artificial neural network composed of multiple building blocks, such as convolution layers, pooling layers, and dense layers (Kim 2014). Various studies have assessed the efficacy of CNN for hate and offensive speech detection (Badjatiya et al. 2017; Alshalan and Al-Khalifa 2020; Zhou et al. 2020).
- Bidirectional Encoder Representations from Transformers (**BERT**), a model pretrained on the English language

Table 4 Data statistics

Value	hb_hate	hb_off	hasoc	olid	heval
Total tweets	5,593	23,353	4,522	14,100	13,000
Total harmful tweets	1,430	19,190	2,279	9,460	7,530
Avg length	20.4	20.03	21.35	28.55	27.24
Labels	Hate; neutral	Offensive; neutral	Hate, offensive, and profane; neutral	Offensive; not offensive	Hate; neutral
% of tweets containing digits	47.72	44.43	42.44	14.06	48.35
% of tweets containing URLs	18.97	12.19	30.38	17.01	37.83
% of tweets containing mentions	61.85	56.99	72	89.51	41.79
% of tweets containing emoji	0	0	14.2	9.74	6.64
% of tweets containing hashtags	14.45	7.55	5.26	15.33	29.08
% of tweets containing emoticons	2.11	1.5	0.73	0.57	0.48
% of tweets containing profane words	30.38	70.72	44.76	18.7	42.13
Representation of mentions	Original	Original	Original	@USER	Original
Representation of URLs	Original	Original	Original	URL	Original

using a masked language modeling objective (Devlin et al. 2019). This model has achieved great success in many natural language processing tasks, including text classification and hate speech detection (for example, Caselli et al. 2021; Li etal. 2021).
- Robustly optimized BERT approach (**RoBERTa**), the model has the same architecture as BERT, but uses a byte-level byte pair encoding as a tokenizer and a different pretraining scheme (Liu et al. 2019). Many researchers utilized RoBERTa for hate speech detection, in particular, in Glazkova et al. (2021), Wiedemann et al. (2020), and Alonso et al. (2020).

In the process of forming a set of models, various classification methods were used that provide methodological diversity due to different approaches that underlie them (statistical models, support vector machines, decision trees, neural networks). The selected traditional machine learning models showed high performance during the comparison of algorithms in related works [in particular, in Krouska et al. (2016), Jianqiang and Xiaolin (2017)]. CNN, BERT, and RoBERTa are widely used solutions for text classification. For example, most of the models presented in Zampieri et al. (2020) were based on BERT- or RoBERTa-style transformers. Among the classic neural network architectures, CNN was the most popular.

To train LR, RF, and LSVC, a bag-of-words model for 10,000 features was built. LR, RF, and LSVC are implemented using Scikit-Learn (Pedregosa et al. 2011). The default settings were utilized for implementation. The parameters of the traditional classifiers are listed in “Appendix B.”

CNN was implemented using Keras (https://github.com/fchollet/keras) with a batch size equal to 256, a number of filters of 256, and a weight decay of 1e−4. The model plot is shown in “Appendix C” (Fig. 3). The parameters of CNN were selected using grid search on the example of the HASOC2020 dataset with baseline preprocessing techniques: translating to lowercase, lemmatization, and removing special characters. The range of the parameters for grid search is presented in “Appendix C” (Table 14). CNN was trained for 100 epochs; however, the actual training time is considerably shorter since strict early stopping was used.

Training CNN was conducted using the FastText word vectors constructed on the texts from Wikipedia 2017, the UMBC WebBase corpus, and the statmt.org news dataset (Joulin et al. 2016). The vector size of FastText’s model is 300. We chose the FastText word vectors as this is a widespread approach to generating word representation for different text mining tasks, especially sentiment analysis and hate speech detection. The authors of Kaibi and Satori (2019) showed the effectiveness of FastText representation compared to Word2Vec (Mikolov et al. 2013) and GloVe (Pennington et al. 2014) for Twitter datasets for sentiment analysis using six machine learning algorithms.

To implement BERT and RoBERTa, we used BERT-base-uncased[7] and RoBERTa-base,[8] respectively, as well as Simple Transformers (Rajapakse 2019) and PyTorch (Paszke et al. 2019). Each model was fine-tuned for two epochs with a maximum sequence length of 128, a learning rate of 4e-5, and a training batch size of 8. We used the AdamW optimizer (Loshchilov and Hutter 2018) with the following parameters: an epsilon hyperparameter of 1e−8; and coefficients used for computing running averages of gradient and its square of (0.9, 0.999).

3.4 Evaluation metrics

Since we used unbalanced datasets in this study, the results were evaluated in terms of the weighted-average F1-score (F1). The weighted-average F1 is calculated by taking the mean value of all per-class F1 while considering the number of true instances for each label.

To obtain more reliable results, we performed fivefold cross-validation for traditional supervised methods (LR, RF, and LSVC) and threefold cross-validation for neural models. We used a small number of folds for neural models due to constraints on machine computing capacity. However, the division into folds was performed with a fixed random seed. Thus, a comparison of the efficiency of techniques for the model was performed on the same test data. All metrics are calculated as mean values for all folds. The values of F1 are presented in the body of the paper, while the values of standard deviation across the folds are shown in “Appendix D.”

3.5 Baselines

We used the following preprocessing for baselines:

- Traditional models (LR, RF, and LSVC) and CNN: translating to lowercase, lemmatization, and removing special characters. Lemmatization was performed using NLTK WordNet Lemmatizer;
- BERT: lowercase;
- RoBERTa: raw text.

In the next section, we evaluate adding and removing basic techniques compared to baseline preprocessing.

[7] https://huggingface.co/bert-base-uncased.

[8] https://huggingface.co/roberta-base.

4 Results and discussion

4.1 Evaluation of basic techniques

In Table 5, we consistently evaluated adding and removing basic techniques compared to baseline preprocessing in terms of F1-score. The table also indicates the values of standard deviation across the folds. For example, according to 3.5, LR used the following baseline preprocessing: translating to lowercase, lemmatization, and removing special characters. *LR baseline − (1)* indicates the LR model with baseline preprocessing, with the exception of converting to lowercase. *LR baseline − (2)* is the LR model with baseline preprocessing but without lemmatization. *LR baseline + (3)* indicates LR with baseline preprocessing but using stemming instead of lemmatization. *LR baseline − (4)* is the LR model with translating to lowercase and lemmatization but without removing special characters. The values that outperform baselines are shown in bold.

The table demonstrates that lemmatization, translating to lowercase, and removing special characters are helpful for traditional machine learning models. Stemming can also help to increase the performance of traditional machine learning models, and for most datasets, it works better than lemmatization.

For CNN, the effect of lemmatization and translation to lowercase is not evident. In general, stemming performs worse than lemmatization for CNN. Removing special characters increases the performance of CNN for most corpora.

As for transformer-based models, the effect of lowercasing expectedly depends on what text preprocessing has been used for model pretraining. Since we used BERT-base-uncased and RoBERTa-base which is the cased model, BERT performs better using lowercasing and RoBERTa in most cases shows higher results if lowercasing was not used. Lemmatization does not significantly affect the performance. Removing special characters and stemming do not mainly increase the results for either models.

4.2 Separate evaluation for techniques

We estimated the performance of each model on all corpora, consistently adding one of the preprocessing techniques to the baseline model. The evaluation results in terms of the weighted-average F1-score are presented in Tables 6 and 7. The first rows of Tables 6 and 7 show the performance of the baseline models, i.e., the models trained on data preprocessed by the techniques specified in Sect. 3.1.5. The following rows demonstrate the performance of the models trained on the texts preprocessed using the baseline techniques and one additional technique from the list of techniques (techniques 5–26 listed in Table 3). Thus, we evaluated the effectiveness of preprocessing techniques one at a time.

In Tables 6 and 7, the scores for tokenizing URLs (9) and tokenizing mentions (11) are missed for the OLID dataset because users' mentions and URLs are already presented in a tokenized form in this corpus. The scores for preprocessing emoji (12–14) are missed for HateBase since this dataset does not contain emoji. The values exceeding the baseline are shown in bold. The standard deviation values across the folds are demonstrated in "Appendix D."

Based on the overall results, we formed five categories depending on the performance of the techniques on different corpora (the dataset level, Table 8). For example, converting emoticons to words (17) and decontraction (22) increase the performance of baselines for most of the models under consideration (in four cases out of six) on HateBase (hb_hate). We have also visualized the effectiveness of preprocessing techniques for each model (the model level, Table 9). For instance, tokenizing mentions (11) and emoticons (16) improved the results of the BERT baseline on all the datasets used. In the next subsections, we discuss the performance of each type of preprocessing techniques.

4.2.1 Handling of digits

Removing digits (5) degrades the performance on most of the datasets and for most of the models. For RF, this technique worsens the scores on all the corpora. For LR, the performance is increased for three out of five datasets. However, the increments are slight (0.01% on HASOC2020, 0.02% on OLID, and 0.26% on HatEval), and therefore they can be random.

Tokenizing digits (6) looks quite effective on HateBase (hb_off), OLID, and HatEval (four models out of six on each datasets). This technique shows higher scores in comparison with the baselines for three models, i.e., in half of the cases. Overall, the effect of tokenizing digits does not have a pronounced character in our experiments.

The results for converting digits to words (7) are quite broadly similar to the results of the previous technique. For OLID and HatEval, the scores are increased for most of the models (five out of six for OLID, four for HatEval). For HatEval, the performance is increased for half of the models. In the case of other datasets, the reported scores were below baselines. However, it is worth noting that converting digits to words in most cases entails performance growths for transformer-based models (four out of five datasets for BERT and three for RoBERTa). For other models, the scores are worsened for the majority of corpora. Probably, it can be related to the fact that pretrained language models are more focused on understanding words than numbers (Wallace et al. 2019; Rogers et al. 2020).

Table 5 Basic techniques, F1 (%). The indices of the techniques are listed in Table 3

Model	hb_hate	hb_off	hasoc	olid	heval
LR					
Baseline	93.02 ± 0.99	96.01 ± 0.54	87.97 ± 1.01	75.44 ± 0.54	64.92 ± 8.95
− (1)	92.97 ± 0.88	95.9 ± 0.56	87.81 ± 1.21	75.04 ± 0.53	64.76 ± 9.04
− (2)	92.34 ± 1.49	95.87 ± 0.54	87.63 ± 1.39	74.84 ± 0.62	**64.98 ± 8.84**
+ (3)	92.94 ± 1.25	**96.13 ± 0.6**	**88.43 ± 0.96**	**75.5 ± 1**	**65.37 ± 8.51**
− (4)	92.95 ± 1.11	95.88 ± 0.61	**87.99 ± 0.99**	74.99 ± 0.58	**64.93 ± 9.04**
RF					
Baseline	92.74 ± 0.62	93.9 ± 0.81	86.56 ± 1.06	74.62 ± 0.61	61.77 ± 13.79
− (1)	91.92 ± 0.25	93.63 ± 0.79	86.09 ± 1.35	74.3 ± 0.49	**61.86 ± 14.38**
− (2)	91.88 ± 0.71	93.09 ± 0.92	**86.59 ± 1.16**	74.22 ± 0.51	**61.83 ± 14.49**
+ (3)	92.58 ± 1.17	**93.91 ± 0.69**	**87.2 ± 0.86**	**75.08 ± 0.37**	**61.83 ± 14.26**
− (4)	92.43 ± 0.96	93.3 ± 0.75	86.49 ± 1.55	74.48 ± 0.34	61.47 ± 14.52
LSVC					
Baseline	92.64 ± 0.77	95.19 ± 0.48	87.05 ± 0.89	73.58 ± 0.77	63.86 ± 6.83
− (1)	**92.73 ± 0.87**	95.05 ± 0.54	86.59 ± 0.79	73.28 ± 0.76	63.61 ± 6.58
− (2)	92.12 ± 1.01	95.15 ± 0.51	86.87 ± 1.3	73.05 ± 0.68	63.02 ± 6.68
+ (3)	92.29 ± 0.82	95.13 ± 0.51	**87.43 ± 1.12**	**74.23 ± 0.97**	**63.89 ± 6.92**
− (4)	92.43 ± 0.64	95.02 ± 0.62	86.96 ± 0.74	73.4 ± 0.91	63.32 ± 6.93
CNN					
Baseline	92.79 ± 1.16	95.25 ± 0.44	88.44 ± 0.22	77.57 ± 0.87	65.99 ± 5.97
− (1)	92.71 ± 1.39	**95.42 ± 0.55**	88.1 ± 0.25	**78.4 ± 0.43**	**66.31 ± 6.27**
− (2)	**92.9 ± 1.25**	**95.54 ± 0.62**	88.26 ± 0.09	**77.61 ± 1.56**	65.72 ± 6.69
+ (3)	**92.82 ± 0.81**	93.12 ± 0.75	87.76 ± 0.74	76.48 ± 0.65	63.92 ± 5.4
− (4)	92.76 ± 1.35	**95.52 ± 0.55**	88.29 ± 0.27	77.36 ± 0.6	64.87 ± 6.5
BERT					
Baseline	93.86 ± 1.13	96.63 ± 0.36	89.72 ± 0.53	80.75 ± 0.42	70.14 ± 7.51
− (1)	93.77 ± 0.87	96.35 ± 0.53	83.89 ± 1.09	79.35 ± 0.3	68 ± 6.19
+ (2)	93.58 ± 0.4	**96.64 ± 0.44**	**90.34 ± 0.44**	80.51 ± 0.38	**70.26 ± 6.41**
+ (3)	93.55 ± 0.62	**96.66 ± 0.65**	**90.16 ± 0.38**	79.77 ± 0.04	69.34 ± 5.97
+ (4)	93.75 ± 0.82	96.53 ± 0.49	90.14 ± 0.55	80.47 ± 0.21	**70.71 ± 6.58**
RoBERTa					
Baseline	93.53 ± 0.79	96.67 ± 0.45	89.77 ± 0.7	80.04 ± 0.67	70.09 ± 7.27
+ (1)	92.42 ± 1.13	96.63 ± 0.43	**89.89 ± 0.51**	**80.49 ± 0.41**	69.39 ± 7.64
+ (2)	**93.92 ± 1.12**	96.3 ± 0.46	**90.23 ± 0.27**	**80.41 ± 0.31**	69.65 ± 6.84
+ (3)	93.13 ± 1.13	96.51 ± 0.49	**89.87 ± 0.54**	79.94 ± 0.21	67.69 ± 6.43
+ (4)	**93.64 ± 1.17**	96.52 ± 0.56	89.39 ± 0.57	79.93 ± 0.34	67.2 ± 7.98

The values that outperform the baselines are shown in bold

4.2.2 Handling of URLs

Removing URLs (8) shows an improvement for all datasets in the case of OLID. The URLs in OLID are replaced with a token *URL*, so in this case we compare the effect of removing and tokenizing URLs. For three datasets (HateBase (hb_hate), HASOC2020, and HatEval), there is a marked deterioration in performance for the greater part of models. Removing URLs also demonstrates ambiguous results from the position of model evaluation. The results are improved on the majority of datasets for three models (RF, BERT, and RoBERTa) and also worsened for most of the corpora for the three other models (LR, LSVC, CNN).

Tokenizing URLs (9) does not achieve high results. This technique reduces scores in most cases for all datasets. The results obtained among the models are similar. The performance growth is detected only for BERT (on three out of five datasets).

Table 6 Separate evaluation of techniques for logistic regression, random forest, and linear support vector classifier, F1 (%)

Model	hb_hate			hb_off			hasoc			olid			heval		
	LR	RF	LSVC	LR	RF	LSVC	LR	RF	LSVC	LR	RF	LSVC	LR	RF	LSVC
bl	93.02	92.74	92.64	96.01	93.9	95.19	87.97	86.56	87.05	75.44	74.62	73.58	64.92	61.77	63.86
5	92.83	92.53	92.51	95.99	93.89	**95.22**	**87.98**	86.53	86.89	**75.46**	74.39	73.58	**65.18**	61.52	63.75
6	92.93	92.65	92.47	**96.03**	93.78	**95.28**	87.96	86.55	**87.12**	**75.48**	**74.77**	**73.72**	**65.13**	**61.82**	**63.89**
7	92.86	92.13	92.44	95.99	93.1	95.11	87.9	86.49	86.7	**75.52**	**74.65**	**73.81**	64.92	**61.85**	**64.01**
8	92.93	**93.09**	92.63	96.01	**94.03**	95.14	87.9	86.47	**87.21**	**75.57**	**74.87**	**73.64**	**65.08**	61.37	63.65
9	92.98	**92.97**	92.52	96	93.85	95.16	87.94	86.56	87.05	–	–	–	64.9	61.3	63.83
10	**93.09**	92.49	92.49	95.97	93.57	95.05	87.96	**86.63**	86.83	**75.5**	**74.9**	73.53	**65.01**	**61.86**	63.71
11	93	92.08	92.59	95.99	93.59	95.04	87.88	**87.05**	86.57	–	–	–	**64.98**	61.59	63.69
12	–	–	–	–	–	–	**88.08**	**86.84**	86.74	**75.57**	**74.82**	**73.61**	**64.98**	61.58	63.77
13	–	–	–	–	–	–	**88.06**	**86.97**	86.9	**75.49**	74.48	73.53	**65.03**	**62.09**	**63.92**
14	–	–	–	–	–	–	87.83	**86.87**	86.72	75.4	74.61	**73.59**	**64.93**	61.57	63.81
15	**93.07**	92.59	92.59	95.99	93.82	95.12	87.83	**86.67**	87.05	**75.48**	**74.64**	**73.66**	**64.94**	61.76	**63.89**
16	**93.13**	92.6	**92.73**	**96.02**	93.73	95.15	87.94	**86.94**	**87.07**	**75.52**	**75.05**	**73.61**	**64.96**	61.54	63.85
17	**93.06**	**92.86**	92.64	**96.02**	93.75	**95.21**	87.94	**86.6**	**87.09**	**75.46**	**74.77**	73.51	64.87	**61.96**	**63.89**
18	92.75	92.49	92.53	95.85	93.38	95.09	87.97	**86.76**	87.05	**75.54**	**74.82**	73.54	**65.52**	**63.29**	**63.69**
19	92.7	92.08	92.47	95.84	93.17	95.08	**88.06**	86.53	87.03	75.39	**74.63**	73.57	**65.4**	**62.02**	63.53
20	92.95	92.69	**92.74**	95.93	93.56	95.15	**88.05**	86.25	**87.18**	**75.53**	**74.75**	73.25	**65.4**	61.58	**64.26**
21	92.76	**92.86**	92.41	95.73	**95.06**	94.9	84.62	84.64	82.63	74.83	**74.89**	73.17	64.79	61.2	63.62
22	**93.06**	92.68	92.48	95.98	93.71	95.19	**87.99**	86.39	**87.07**	**75.71**	**74.99**	**73.87**	**65.11**	**62.07**	**64.12**
23	93	**93.08**	92.6	**96.02**	93.76	**95.2**	87.94	**86.65**	87.01	**75.62**	**74.76**	73.55	64.82	**61.89**	**63.91**
24	90.51	91.09	89.63	95.31	**95.16**	93.97	87.46	**87.52**	81.9	74.94	**75.05**	72.38	**66.08**	**64.05**	**64.19**
25	92.95	92.7	**92.66**	**96.04**	93.83	**95.23**	88.1	84.85	86.79	**75.62**	**75.03**	**73.78**	**65.08**	**61.92**	**63.92**
26	**93.13**	92.45	92.56	95.99	93.72	**95.22**	87.92	**86.67**	**87.07**	75.34	**74.84**	73.56	64.77	61.46	63.74

The indices of the techniques are listed in Table 3
The values that outperform the baselines are shown in bold
Bl baseline

4.2.3 Handling of mentions

Removing mentions (10) leads to a performance decrease for all the models on HateBase (hb_off). For HASOC2020, this technique does not show any improvement for the majority of models. For OLID that contains tokenized mentions, the performance increases in most cases. Otherwise, there is no evident effect of removing mentions. The technique fails on all datasets while using LSVC and fails in most cases utilizing all types of neural networks but, more often than not, removing mentions exceeds baseline results for LR and RF.

Tokenizing mentions (11) worsened the results in most cases for all datasets. However, this technique leads to performance increases using BERT (all datasets). The results for RoBERTa are ambiguous (an increase on HateBase (hb_hate) and HASOC2020, a decrease on HateBase (hb_off) and HatEval. For traditional machine learning methods, the technique does not beat baselines in the majority of cases.

4.2.4 Handling of emoji and emoticons

The HateBase dataset does not contain emoji. Therefore, we do not evaluate the techniques related to the handling of emoji on this corpus.

Removing emoji (12) shows mixed results across the datasets (improvement using five models out of six on OLID, three models on HASOC2020, and two models on HatEval). It is successful for LR (the performance growth on all datasets), RF, and BERT (improvement for two datasets out of three). For LSVC, CNN, and RoBERTa, the scores are lower than the baseline in most cases.

Tokenizing emoji (13) increases the performance for the majority of models on HASOC2020 and HatEval but worsens the results on OLID. This technique improves scores on all datasets using LR and RF. For BERT and RoBERTa, tokenizing emoji helps to beat the baseline for two datasets out of three. For LSVC and CNN, the results are decreased in two of three cases.

Table 7 Separate evaluation of techniques for Convolutional Neural Network, BERT, and RoBERTa (RB), F1 (%)

Model	hb hate			hb off			hasoc			olid			heval		
	CNN	BERT	RB	CNN	BERT	RB	CNN	BERT	RB	CNN	BERT	RB	CNN	BERT	RB
bl	92.79	93.86	93.53	95.25	96.63	96.67	88.44	89.72	89.77	77.57	80.75	80.04	65.99	70.14	70.09
5	**93.18**	93.42	**93.56**	**95.44**	**96.73**	96.46	87.42	89.44	89.48	77.3	80.51	**80.18**	64.81	**70.68**	69.17
6	92.46	**94.29**	**94.19**	**95.45**	**96.72**	96.66	87.91	89.63	89.17	**77.64**	80.06	80	64.43	**70.88**	**70.51**
7	**92.8**	**94.08**	**93.68**	95.2	96.6	**96.68**	87.56	**90**	89.54	**77.87**	**80.78**	79.58	63.84	**70.94**	**70.3**
8	92.67	93.29	**93.79**	**95.46**	**96.65**	**96.68**	88	**89.81**	89.7	**77.61**	**80.89**	**80.27**	63.54	**71.01**	69.86
9	92.59	93.77	93.28	**95.37**	**96.72**	96.56	**88.58**	**90.18**	89.55	–	–	–	62.95	**70.49**	70.08
10	92.66	**94.02**	**93.6**	94.99	96.52	96.6	88.03	89.7	89.68	**77.74**	80.49	**80.54**	65.34	**70.96**	69.59
11	92.05	**93.97**	**93.65**	**95.42**	**96.85**	96.55	88.14	**89.9**	**90.25**	–	–	–	63.43	**71.05**	69.54
12	–	–	–	–	–	–	88.17	**90.2**	89.39	**78.16**	80.62	**80.76**	63.63	**71.34**	69.28
13	–	–	–	–	–	–	**88.61**	**89.85**	**89.85**	76.13	80.69	**81.1**	64.49	**70.95**	69.2
14	–	–	–	–	–	–	88.02	**89.98**	89.7	**78.25**	80.74	**80.23**	64.07	**70.68**	69.43
15	92.12	93.6	**93.81**	**95.44**	**96.64**	96.65	88.33	**90.27**	**89.88**	**78.17**	80.54	**80.15**	**66.27**	**70.83**	**70.22**
16	92.47	**93.95**	93.46	**95.46**	**96.75**	96.59	87.56	**89.87**	89.26	77.17	**80.77**	78.9	64.69	**70.8**	**70.51**
17	**92.85**	**94.21**	93.44	**95.52**	**96.78**	96.59	88.18	89.5	**89.98**	77.42	79.98	**80.54**	**66.41**	**70.61**	68.87
18	**92.86**	**95.89**	**93.89**	95.2	96.49	96.49	87.78	**89.92**	89.76	**78.43**	80.36	**80.28**	62.97	**70.87**	69.77
19	92.29	**94.23**	93.09	95.12	96.58	96.51	88.28	**89.94**	89.67	77.29	80.17	**80.26**	64.79	**70.69**	**72.59**
20	92.54	93.67	93.23	**95.61**	**96.69**	95.74	88.25	**90.23**	89.39	76.97	80.36	76.69	64.29	70.05	68.48
21	92.71	93.72	93.24	94.85	96.59	96.44	84.21	**90.12**	**89.9**	77.37	79.42	**80.12**	64.88	68.92	66.11
22	**92.85**	**96.67**	**93.99**	**95.66**	**96.67**	96.58	88	89.7	89.77	**77.83**	80.45	**80.55**	65.79	**70.4**	67.93
23	**93.09**	93.74	93.36	**95.54**	**96.69**	96.63	87.94	**89.85**	89.68	77.31	80.39	**80.1**	**66.47**	**70.91**	70.02
24	90.58	**94.54**	90.83	94.92	**96.73**	95.46	87.04	**90.05**	88.88	77.21	80.07	79.43	**66.14**	**70.96**	68.41
25	92.78	93.49	92.98	**95.59**	96.62	95.71	88.28	89.63	89.45	**77.71**	80.4	**80.55**	64.84	**70.31**	67.82
26	92.47	93.48	**94.01**	**95.49**	**96.65**	**96.74**	88.42	**89.94**	**90.12**	77.55	80.44	**80.21**	65.12	**70.9**	69.86

The indices of the techniques are listed in Table 3
The values that outperform the baselines are shown in bold
Bl baseline

Table 8 Effect of techniques per dataset

Results	Datasets				
	hb_hate	hb_off	hasoc	olid	heval
Higher in all	–	–	–	8	–
Higher in most	17, 22	6, 8, 17, 23, 26	13, 26	6, 7, 10, 12, 15, 16, 18, 22, 25	6, 7, 13, 15, 17, 18, 19, 22, 23, 24,25
Equally	7, 10, 16, 18, 23	5, 16, 25	12, 15, 16, 17, 20	14,17, 23	10, 16
Lower in most	5, 6, 8, 9, 11, 15, 19, 20, 21, 24, 25, 26	7, 9, 11, 15, 20, 21, 22, 24	5, 6, 7, 8, 9, 10, 11, 14, 18, 19, 21, 22, 23, 24	5, 13, 19, 20, 21, 24, 26	5, 8, 9, 11, 12, 14, 20, 26
Lower in all	–	10, 18, 19	25	–	21

The indices of the techniques are listed in Table 3

Converting emoji to words (14) generally does not have a significant effect on the performance of the classification. The technique demonstrates the lower performance in most cases for all traditional approaches, CNN, and RoBERTa. For BERT, the performance is slightly improved on HASOC2020 and HatEval.

The proportion of tweets containing emoticons is not large. The largest proportion of emoticons is present in the HateBase dataset (2.11% for hb_hate and 1.5% for hb_off). For the rest of the datasets, the proportion of tweets with emoticons is less than one percent.

Table 9 Effect of techniques per model

Results	Models					
	LR	RF	LSVC	CNN	BERT	RoBERTa
Higher in all	12, 13	13	–	–	11, 16	–
Higher in most, or equally*	5, 6, 10, 15, 16, 17, 20, 22, 25	8, 10, 12, 17, 18, 21, 23, 24	6, 16, 17, 20, 22, 25	9*, 15, 17, 22, 23	6, 7, 8, 9, 12, 13, 14, 15, 17, 18, 19, 22, 23, 24, 26	7, 8, 11*, 13, 15, 26
Lower in most	7, 8, 11, 14, 18, 19, 23, 26	6, 7, 9, 11, 14, 15, 16, 19, 20, 22, 25, 26	5, 7, 8, 12, 13, 14, 15, 18, 23, 26	5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 18, 20, 24, 25, 26	5, 10, 20, 21, 25	5, 6, 10, 12, 14, 16, 17, 18, 19, 22, 23, 25
Lower in all	9, 21, 24	5	9, 10, 11, 19, 21, 24	19, 21	–	9, 20, 21, 24

Since the number of datasets is odd, an equal number of performance growths and falls is possible only for tokenizing URLs and mentions (techniques 9 and 11) that are not evaluated for the OLID dataset. These cases are marked with an asterisk (*). The indices of the techniques are listed in Table 3

In our experiments, removing emoticons (15) shows an improvement in most cases for OLID and HatEval, produces mixed results for HASOC2020, and demonstrates the deterioration of the scores for HateBase. Despite the ambiguous results at the dataset level, this technique shows itself well for both transformers (improvement for three and four datasets for BERT and RoBERTa, respectively) and LR (for three datasets). For the two remaining methods, the removal of emoticons demonstrates a deterioration in most cases.

Tokenizing emoticons (16) does not have a strong influence on the results. For most datasets and models, the results are slightly improved or worsened equally. However, for BERT, the tokenization of emoticons causes a small increase across all datasets (from 0.02% to 0.66%).

Converting emoticons to words (17) shows an improvement in most or half of the models across all datasets. The technique also demonstrates an improvement of the scores in most cases for all models but RoBERTa. For RoBERTa, the results are improved only in two cases out of five.

In general, the significance of emoticons as features for the task of hate and offensive speech detection looks higher than the significance of emoji. Apparently, this is due to the fact that emoticons more often carry a pronounced sentiment. At the same time, emoji are often used to express a wider range of emotions, including fear, surprise, joy, and so on Guibon et al. (2016).

4.2.5 Handling of hashtags

Removing hashtags (18) leads to ambiguous results among the datasets. The performance decreases on HateBase (hb_off) for all models and on HASOC2020 for most models. For OLID, the scores are improved in most cases. For HateBase (hb_hate), the performance decreases for half of models. At the model level, the results are also ambiguous, but in general in most cases, this technique negatively influences the results.

Tokenizing hashtags (19) shows an absolute deterioration on the HateBase (hb_off) dataset. For the rest of the datasets, the technique does not have a significant impact on the results. At the model level, the tokenization of hashtags demonstrates a deterioration for all datasets when using LSVC and CNN. In general, the technique rather worsens the results.

Hashtag segmentation (20) also has a negative impact on the results for most models and datasets. Less deterioration is demonstrated for HASOC2020 (a decrease in performance in 50% of cases). However, HASOC2020 contains only 6% of tweets with hashtags, which is the smallest proportion among all datasets. At the model level, the results are improved for three datasets out of five using LR and LSVC. RoBERTa shows a deterioration across all datasets. For other models, the technique mainly worsens the results.

In our experiments, the processing of hashtags shows a negative impact on the performance of the models for hate and offensive speech detection. This technique does not improve the results on any dataset. That is, hashtags are generally an important and indivisible part of the text, and not as a set of meanings of the hashtag segments. However, for some traditional models (for example, LR and LSVC), improvement can be obtained by segmentation of hashtags.

4.2.6 Lexical transformations

Removing stopwords (21) demonstrates lower performance in most cases for all datasets. For HatEval, the results are worsened across all models. This technique shows some of the worst results in our experiments. A slight improvement (on three of the five datasets) is obtained only for RF. For LR, LSVC, CNN, and RoBERTa, the results are worsened for all datasets. This effect can be associated with a short tweet length. Similar results were obtained in Baruah et al. (2019), where the removal of the stopwords worsened the performance of BiLSTM. This technique also negatively affected the performance of decision trees and naive Bayes classifiers in Saeed et al. (2022) and logistic regression and support vector classifiers in Garouani et al. (2021). The authors of Do et al. (2019) noted that it is not necessary to remove all stopwords to detect hate speech in social media texts because having a few stopwords affects the results.

Decontraction (22) and replacing acronyms (23) have no evident effect on the performance. Decontraction leads to improvement in most cases for HateBase (hb_hate), OLID, and HatEval. Replacing acronyms increases the scores for HateBase (hb_off) and HatEval in most cases and for HateBase (hb_hate) and OLID in half of the cases. Both techniques worsen the results on HASOC2020. Decontraction also is not helpful for HateBase (hb_off). For each of these techniques, the performance of one-half of the models is generally improved, and the performance of other models is worsened.

Tokenizing profanity (24) shows poor results in our experiments. For four out of five datasets, the scores are deteriorated for most models. For LR, LSVC, and RoBERTa, this technique leads to degradation on all datasets.

4.2.7 Corrections

The effectiveness of spelling correction (25) varies for different datasets. For OLID and HatEval, it improves the results for most models while for HASOC2020, the scores are worsened across all models. The technique generally improves the results of LR (three datasets) and LSVC (four datasets), but for other models, it leads to a worsening of scores for most datasets.

Removing repetitions (26) mostly improves the results on HatBase (hb_off) and HASOC2020, but it is not effective on other datasets. The technique is impactful for BERT (three datasets) and RoBERTa (four datasets) and leads to a general performance degradation on for other models.

4.2.8 Summary for separate evaluation of preprocessing techniques

The choice of preprocessing techniques for hate and offensive speech detection in Twitter texts strongly depends on the specifics of the dataset and the applied model. However, our experiments on separate evaluation of preprocessing techniques allowed us to draw the following conclusions.

Mainly, the handling of digits has no positive effect on the classification performance. The impact of tokenizing digits and converting digits to words seems mixed. However, converting digits to words positively affects the performance of transformer-based models. In our experiments, removing URLs performs better than their tokenization. Probably this suggests that the presence of URLs is not associated with the presence of hate and offensive speech. For transformer-based models, removing URLs generally performs well. Based on our experiments, mentions should not be removed. It is important for solving hate and offensive speech detection tasks that the tweet is addressed to someone. Tokenizing mentions works poorly for traditional methods. However, for transformers, tokenization generally improves the baseline results. Removing and tokenizing emoji and emoticons produce no clear effect on the performance of text classification. In general, converting emojis to words worsens the results. On the contrary, by converting emoticons into words, we mainly achieve positive results. Probably, this is because emoticons usually express sentiment while emoji show a wide range of different feelings. The removal and tokenization of hashtags do not have a strong effect. We also assume that hashtags are important as complete semantic units because their segmentation worsens the results in our experiments. As regards lexical transformations, profanity tokenization and removal of stopwords negatively impact the performance of hate and offensive speech detection in Twitter texts. Our experiments show that the meaning of profane words is important for classification, so we cannot replace them with tokens. Removing stopwords shows poor results since tweets are texts of low length and stopwords represent their important semantic components. Further research might explore the impact of the proportion of removed stopwords. Decontraction and replacing acronyms demonstrate no benefit for hate and offensive speech detection. The results of spelling correction depend on the model used. This technique improves the results for two traditional methods that used bag-of-words text representations (LR and LSVC). Removing repeated letters helps BERT and RoBERTa to increase the scores improving the tokenization performance for these models.

In our experiments, we did not notice the influence of the type of harmful content (hate or offensive) on the effectiveness of certain preprocessing techniques. However, generally the removal of URLs positively influenced the performance of offensive speech detection (HateBase (hb_off) and OLID). For hate speech, its influence was mainly negative. For both offensive speech datasets, tokenizing digits mostly positively affected the performance while tokenizing hashtags, hashtag segmentation, removing stopwords, and tokenizing profanity had a negative effect. For hate speech datasets (HateBase (hb_hate) and HatEval), converting emoticons to words and decontraction generally had a positive influence on the results while removing digits, removing and tokenizing URLs, tokenizing mentions, hashtag segmentation, removing stopwords, and removing repetitions negatively affected the classification performance.

In Table 10, we ranked all preprocessing techniques based on the average values of *relative growths*. Relative growths were calculated as the value of the F1-score growth, expressed as a percentage, relative to the baseline model for each dataset. For example, the LR baseline model showed 93.02% of F1-score on HateBase (hb_hate). The same model with removing digits (5) achieved 92.83%. In this case, the relative growth is $(92.83 - 93.02)/93.02 * 100 = 0.2$. Table 10 ranks techniques in accordance with the average values of relative growths for each pair "model - dataset" (column "All models") and separately for each model (columns "LR," "RF," "LSVC," "CNN," "BERT," and "RoBERTa"). For example, the "LR" column shows the sum of relative growths for all datasets when using logistic regression. We can see that the greatest total relative growth, taking into account all the datasets, is obtained for stemming (3) and the lowest total relative growth is demonstrated for removing stopwords (21). We did not show the results for lowercase (1), lemmatization (2), and removing special characters (4), since these techniques were used for some baselines and were not used for others.

Decontraction (22), removing emoticons (15), replacing acronyms (23), removing hashtags (18), and converting emoticons to words (17) demonstrated the highest relative success across all models. However, the results varied depending on the model used. The top-5 effective techniques for LR were stemming (3), hashtag segmentation (20), decontraction (22), removing hashtags (18), and spelling correction (25). The best results for RF were archived using tokenizing profanity (24), removing hashtags (18), stemming (3), tokenizing emoji (13), and replacing acronyms (23). For LSVC, the highest scores were obtained with stemming (3), decontraction (22), hashtag segmentation (20), tokenizing digits (6), and spelling correction (25). For all traditional methods, stemming showed a fairly high performance. The highest relative success for CNN was achieved using removing emoticons (15), converting emoticons to words (17), replacing acronyms (23), decontraction (22), and spelling correction (25). The top-5 techniques for BERT also included decontraction (22), removing hashtags (18), removing emoji (12), tokenizing mentions (11), and converting digits to words (7). For RoBERTa, the most effective techniques were tokenizing hashtags (19), removing repetitions (26), removing emoticons (15), tokenizing emoji (13), and tokenizing digits (6). For both transformer-based models, tokenizing mentions (11) and emoji (13) and removing emoji (12), emoticons (15), and hashtags (18) were quite effective.

The values of the relative growths are also utilized to visualize the effectiveness of the techniques presented in Figs. 1 and 2. These figures show the performance increases or decreases obtained by a technique for each model. For instance, Fig. 1 demonstrates that the results of LR have worsened on all datasets when removing stopwords (21). For tokenizing profanity (24), the results have worsened for all datasets with the exception of HatEval.

4.3 Combinations of techniques

In this subsection, we attempt to combine the best techniques from the previous step. For this purpose, we use the following ways for technique combination.

- **Individual combination** For each model and dataset, we combine the techniques that outperformed the corresponding baseline during the separate evaluation. If the improvement is achieved using incompatible techniques (for example, removing and tokenizing digits), we choose a technique that has the greatest positive effect. If both techniques showed the same results (for example, techniques 16 and 17 for HateBase (hb_off)), we evaluate both of them.
- **Combination at the model level** At the model level, we combined the techniques that lead to improvement on all, or on most, datasets (see Table 9). If the improvement is achieved using incompatible techniques, we chose a technique that shows an improvement for a larger number of datasets. In the case of an equal number of datasets, we take into account the average positive effect across all datasets.

The lists of the techniques included in technique combinations are presented in Table 11.

The evaluation results for technique combinations are presented in Table 12. The scores that exceed baselines are shown in bold. The asterisk (*) marks absolutely best results for a specific model on a given dataset. Our results show

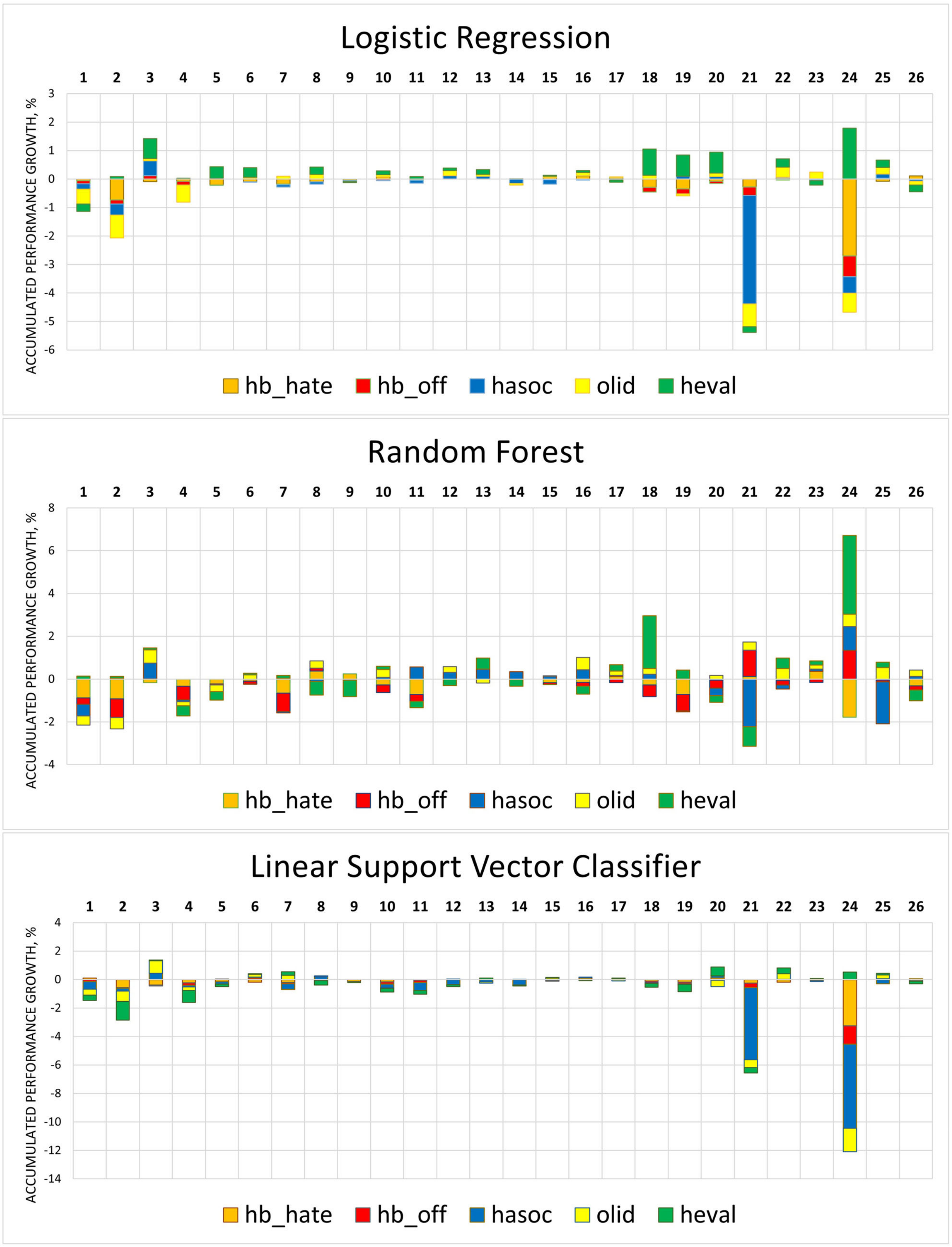

Fig. 1 Stacked bar diagram for relative performance growth across traditional machine learning algorithms: LR, RF, and LSVC. The indices of the techniques are listed in Table 3

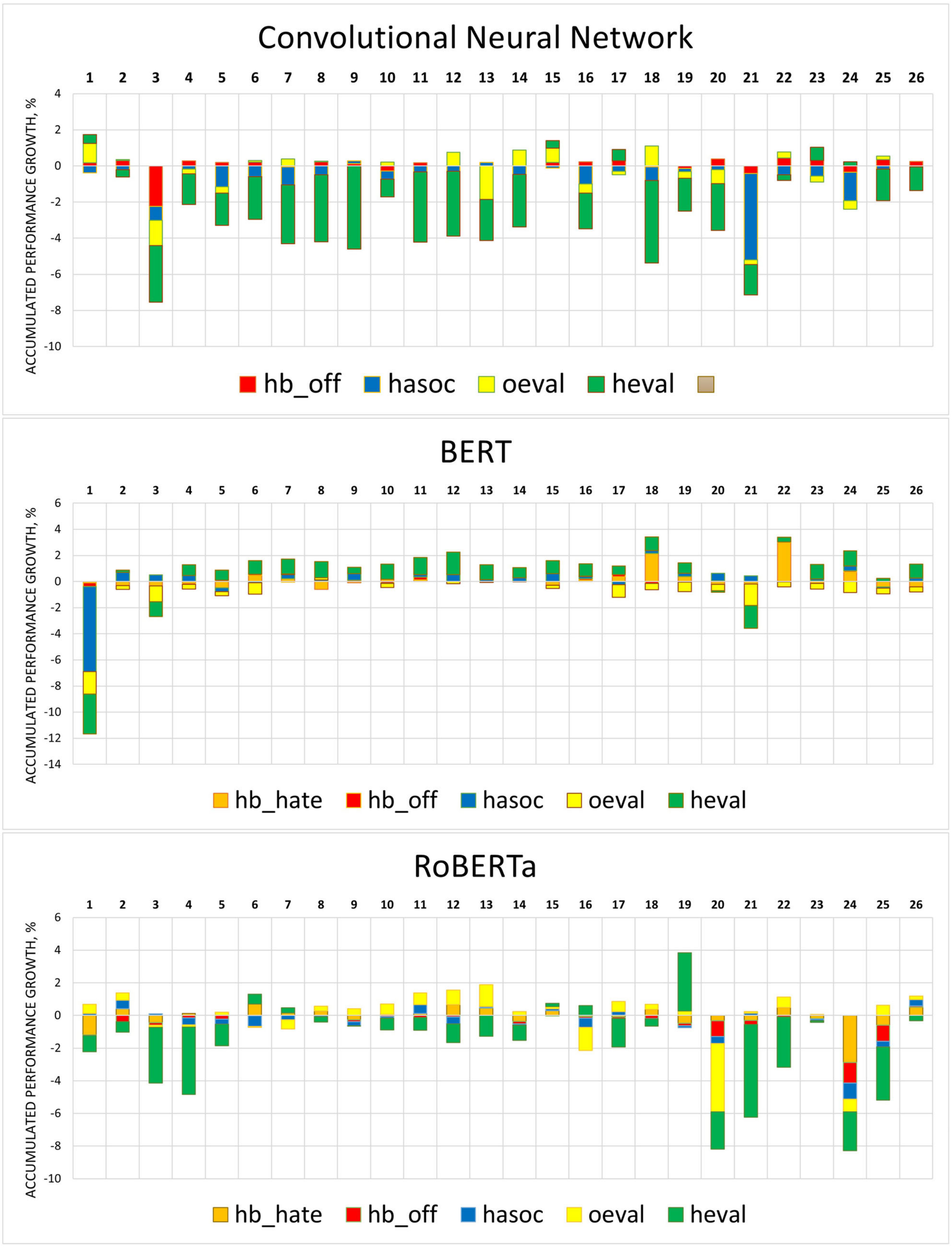

Fig. 2 Stacked bar diagram for relative performance growth across neural models: CNN, BERT, and RoBERTa. The indices of the techniques are listed in Table 3

Table 10 Relative success of the technique (based on the average values of relative growths)

All models	LR	RF	LSVC	CNN	BERT	RoBERTa
22 - decontraction	3	24	3	15	22	19
15 - removing emoticons	20	18	22	17	18	26
23 - replacing acronyms	22	3	20	23	12	15
18 - removing hashtags	18	13	6	22	11	13
17 - converting emoticons to words	25	23	25	25	7	6
12 - removing emoji	12	17	16	26	24	11
19 - tokenizing hashtags	13	22	15	10	16	8
13 - tokenizing emoji	6	16	17	14	13	18
26 - removing repetitions	16	12	23	5	15	12
6 - tokenizing digits	19	8	8	6	14	10
10 - removing mentions	8	6	13	19	9	9
8 - removing URLs	10	14	7	12	8	23
16 - tokenizing emoticons	5	10	9	20	10	7
14 - converting emoji to words	23	15	26	16	23	17
7 - converting digits to words	17	9	14	7	19	14
11 - tokenizing mentions	15	26	12	13	6	16
9 - tokenizing URLs	11	11	5	8	26	5
5 - removing digits	9	20	18	18	17	22
25 - spelling correction	7	5	19	9	20	3
3 - stemming	14	19	10	24	5	25
20 - hashtag segmentation	26	25	11	11	25	21
24 - tokenizing profanity	24	21	21	21	3	20
21 - removing stopwords	21	7	24	3	21	24

The numbers of the techniques are given in the cells. The technique type is highlighted in color in accordance with Table 3. The indices of the techniques are also listed in Table 3, as well as being duplicated in the column "All models" for the convenience of readers

that the combination of techniques that were successful in a separate evaluation does not always give the best effect. This is especially noticeable for transformers when both technique combinations work worse than the baselines trained on raw texts. The combination of successful preprocessing techniques works better for traditional methods. For HateBase (hb_hate), individual combinations of preprocessing techniques show the highest in-dataset results for LR and RF. For HateBase (hb_off), both combinations improve the baseline results for RF. For HASOC2020, the individual combination increases the baseline scores for LR and CNN. For CNN, the combination at the model level also exceeds the baseline. The best result for OLID dataset and RF was obtained using the corresponding combination at the model level. For HatEval, the individual combination shows the best results for RF and LSVC. The baseline result is also improved for LR (both combinations), RF and LSVC (combinations at the model level), and CNN (individual combinations).

5 Limitations

The generalizability of the results is subject to certain limitations. For instance, the messages on Twitter are restricted to 140 symbols. Therefore, the Twitter language is determined by this limitation, and it is usually unstructured and informal (Naseem et al. 2021). For the texts of other genres and other topics, the results obtained can be very different from ours.

In this study, we did not aim to identify the best algorithm for detecting hate and offensive speech in Twitter messages. The research was focused on the evaluation of preprocessing techniques using several machine learning approaches, as well as analysis and summarization of the results. This was the reason why the approaches used in this study were common and standard. Further research might explore additional types of classifiers, such as the naive Bayes classifier or other types of neural networks (long short-term memory, generative adversarial networks, etc.).

During the evaluation of the combinations (Subsection 4.3), we did not evaluate the impact of ordering the techniques within the combination on the performance of the model. This issue can be explored in further research.

In this work, we evaluated the effectiveness of the technique in terms of the presence or absence of an increase in the weighted-average F1-score using several corpora and models. The results are presented as the average values between folds. Additionally, the values of standard deviation are given. An additional study of the statistical significance of the obtained values also may be a direction for further work.

6 Conclusion

This paper investigated the effect of 26 preprocessing techniques for tweet classification using four datasets for hate and offensive speech detection. Each preprocessing tech-

Table 11 Technique combinations

Model	Datasets				
	hb_hate	hb_off	hasoc	olid	heval
LR_{ind}	10, 16, 22, 26	6, 16/17, 23, 25	5, 12, 19, 22	7, 8, 10, 12, 16, 18, 22, 23, 25	5, 8, 10, 13, 16, 18, 22,25
LR_{comb}	6, 10, 12, 16, 20, 22, 25				
RF_{ind}	8, 17, 21, 23	8, 21, 24	11, 13, 16, 18, 23, 24, 26	6, 8, 10, 12, 16, 18, 21, 22, 23, 24, 25, 26	7, 10, 13, 17, 18, 22, 23, 24, 25
RF_{comb}	8, 10, 13, 17, 18, 21, 23, 24				
LSVC_{ind}	16, 20, 25	6, 17, 23, 25, 26	6, 8, 17, 20, 22, 26	7, 8, 14, 15, 22, 25	7, 13, 15/17, 20, 22, 23, 24, 25
LSVC_{comb}	6, 16, 20, 22, 25				
CNN_{ind}	5, 17, 18, 22, 23	6, 8, 11, 17, 20, 22, 23, 25, 26	9, 13	7, 8, 10, 14, 15, 18, 22, 25	17, 23, 24
CNN_{comb}	15, 22, 23				
BERT_{ind}	6, 10, 17, 18, 22, 24	5, 9, 11, 17, 20, 22, 23, 24, 26	7, 9, 11, 12, 15, 20, 21, 23, 24, 26	7, 8, 16	7, 8, 11, 12, 15, 18, 22, 23, 24, 25, 26
BERT_{comb}	7, 9, 11, 12, 16, 18, 22, 23, 24, 26				
$\mathrm{RoBERTa}_{ind}$	6, 8, 11, 15, 18, 22, 26	7, 8, 26	11, 13, 17, 21, 26	5, 8, 10, 13, 17, 18, 21, 22, 23, 25, 26	6, 16, 19
$\mathrm{RoBERTa}_{comb}$	7, 8, 13, 15, 26				

The indices of the techniques are listed in Table 3

Table 12 Results for technique combinations (F1-score, %)

Model	Datasets				
	hb_hate	hb_off	hasoc	olid	heval
LR_{ind}	**93.18*** ± **1.04**	95.72 ± 0.67(16) 95.72 ± 0.67(17)	**88.26** ± **1.13**	74.96 ± 0.45	**65.15** ± **8.44**
LR_{comb}	92.64 ± 1.1	95.61 ± 0.57	84.75 ± 0.93	75.03 ± 0.93	**65.72** ± **8.16**
RF_{ind}	**93.14*** ± **0.88**	**95.25** ± **0.8**	84.8 ± 1.1	74.54 ± 0.6	**64.38*** ± **7.1**
RF_{comb}	90.8 ± 0.74	**95.08** ± **0.76**	85.32 ± 0.9	**75.14*** ± **0.59**	**62.86** ± **10.22**
LSVC_{ind}	92.24 ± 0.97	94.88 ± 0.78	83.2 ± 0.67	73.22 ± 0.55	**64.48*** ± **4.21** (15) **64.47** ± **4.21** (17)
LSVC_{comb}	92.28 ± 0.74	94.84 ± 0.6	83.05 ± 0.65	73.29 ± 0.54	**64.32** ± **5.87**
CNN_{ind}	92.77 ± 0.74	94.77 ± 0.66	**88.64** ± **0.24**	77.45 ± 1.03	**66.26** ± **4.74**
CNN_{comb}	93.11 ± 0.99	95.68 ± 0.66	**87.87** ± **0.05**	77.09 ± 1.31	64.56 ± 6.53
BERT_{ind}	91.58 ± 1.29	95.4 ± 0.87	85.23 ± 0.11	80.3 ± 0.4	69.87 ± 5.68
BERT_{comb}	91.57 ± 0.46	95.64 ± 0.82	87.65 ± 0.16	79.1 ± 0.21	70.06 ± 5.54
$\mathrm{RoBERTa}_{ind}$	92.44 ± 1.45	95.75 ± 0.49	86.21 ± 0.85	78.72 ± 0.65	69.68 ± 7.04
$\mathrm{RoBERTa}_{comb}$	93.29 ± 0.82	96.21 ± 0.5	89.56 ± 0.68	79.35 ± 0.67	67.96 ± 6.1

The values that outperform the baselines are shown in bold nique was evaluated using three traditional machine learning methods (logistic regression, random forest, and linear support vector classifier) and three deep learning methods (convolutional neural network, bidirectional encoder representations from transformers (BERT), and RoBERTa). We used a bag-of-words text representation for traditional methods and FastText for CNN. Both text representations are widely utilized for text classification. In this work, we separately evaluated the techniques and made conclusions about their effectiveness for different datasets and methods. We examined a large number of techniques that have not been evaluated in a comparative study in the past. We divided the preprocessing techniques into categories in accordance with their effectiveness at the model and dataset levels and ranked them based on the relative success across the datasets. We demonstrated that some techniques provided better results in classification for some models, while others decreased the scores. Similar to previous research, our results showed that the effectiveness of text preprocessing techniques varies across different datasets and methods. Therefore, the choice of the method for classifying texts of a particular dataset is a crucial step for hate and offensive speech detection.

Combining preprocessing techniques can also produce different results. We explored two ways to combine successful techniques. In general, technique combinations performed better for traditional methods than for deep learning methods. For transformer-based models, the results obtained using technique combinations were the worst among all models.

In future studies, we will investigate these techniques in different domains and explore other ways to construct effective combinations of preprocessing techniques. A further study could also assess higher-level preprocessing techniques, such as identifying synonyms and other semantic relations, as well as more detailed lexical features, such as detecting urban language and jargon. Another important issue for future research is testing these techniques on fine-grained datasets for hate and offensive speech detection.

Author Contributions The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

Declarations

Conflict of interest The authors declare no competing interests.

Appendix A: Lists of emoticons, contractions, and acronyms

- Converting emoticons to words (17)::D,)),:),:)), etc. → *happy*; ((,:(,:((, etc. → *sad*; XD → *laugh*;:* → *kiss*.
- Decontraction (22): *won't* → *will not*; *shan't* → *shall not*; *can't* → *can not*; *couldn't* → *could not*; *'re* → *are*; *'s* → *is*; *'d* → *would not*; *'ll* → *will not*; *'ve* → *have*; *'m* → *am*.
- Replacing acronyms (23): *?4U* → *question for you*, *2moro* → *tomorrow*, *2nte* → *tonight*, *asap* → *as soon as possible*, *atm* → *at the moment*, *awsm* → *awesome*, *b2w* → *back to work*, *bff* → *best friends forever*, *brb* → *be right back*, *btw* → *by the way*, *cu* → *see you*, *cus* → *see you soon*, *deti* → *don't even think it*, *diky* → *do i know you*, *g2g* → *got to go*, *gr8* → *great*, *idc* → *i don't care*, *idk* → *i don't care*, *lmk* → *let me know*, *lol* → *laughing out loud*, *lu* → *love you*, *ly* → *love you*, *myob* → *mind your own business*, *omg* → *oh my god*, *pls* → *please*, *plz* → *please*, *rip* → *rest in peace*, *rofl* → *rolling on the floor*, *sep* → *someone else's problem*, *sup* → *what's up*, *tnx* → *thanks*, *ttyl* → *talk to you later*.

Appendix B: The main parameters of LR, RF, and LSVC

See Table 13.

Table 13 Parameters for LR, RF, and LSVC

Model	Parameters
LR	Penalty—L2; the tolerance for stopping criteria—1e−4; the algorithm to use in the optimization problem—LBFGS; the maximum number of iterations taken for the solvers to converge—100
RF	The number of trees—100; criterion—the Gini index; no limit on the maximum depth of the tree; the minimum number of samples required to split an internal node—2; the minimum number of samples required to be at a leaf node—1
LSVC	Penalty – L2; loss—the squared hinge loss; the tolerance for stopping criteria—1e−4; the maximum number of iterations taken for the solvers to converge—100

Table 14 Parameters for grid search

Parameter	Range	Chosen value
Number of convolutional layers	[1;3]	2
Batch size	[32,64, 128, 256]	256
Number of filters	[32, 64, 128, 256, 512]	256
Kernel size	[2;3]	2
Dropout	[0.1,0.9], step = 0.1	0.5

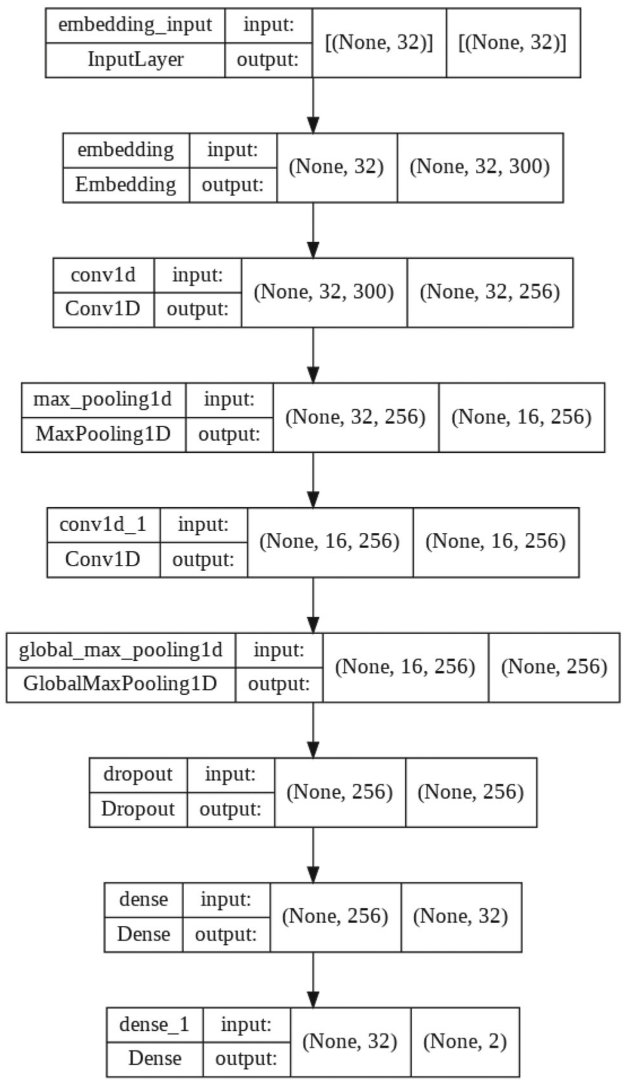
embedding_input | InputLayer | input: | output: | [(None, 32)] | [(None, 32)]
embedding | Embedding | input: | output: | (None, 32) | (None, 32, 300)
conv1d | Conv1D | input: | output: | (None, 32, 300) | (None, 32, 256)
max_pooling1d | MaxPooling1D | input: | output: | (None, 32, 256) | (None, 16, 256)
conv1d_1 | Conv1D | input: | output: | (None, 16, 256) | (None, 16, 256)
global_max_pooling1d | GlobalMaxPooling1D | input: | output: | (None, 16, 256) | (None, 256)
dropout | Dropout | input: | output: | (None, 256) | (None, 256)
dense | Dense | input: | output: | (None, 256) | (None, 32)
dense_1 | Dense | input: | output: | (None, 32) | (None, 2)

Fig. 3 CNN architecture

Appendix C: The architecture of CNN

See Table 14 and Fig. 3.

Appendix D: The values of standard deviation across the folds for separate evaluation of preprocessing techniques

See Tables 15 and 16.

Table 15 Separate evaluation of techniques for logistic regression, random forest, and linear support vector classifier, standard deviation across the folds (%)

Model	hb_hate			hb_off			hasoc			olid			heval		
	LR	RF	LSVC	LR	RF	LSVC	LR	RF	LSVC	LR	RF	LSVC	LR	RF	LSVC
Baseline	0.99	0.62	0.77	0.54	0.81	0.48	0.01	1.06	0.89	0.54	0.61	0.77	8.95	13.79	6.83
5	1.12	0.49	0.98	0.56	0.67	0.43	1.13	1.32	0.62	0.47	0.29	0.64	8.7	14.18	6.34
6	1.05	0.89	0.73	0.56	0.63	0.47	0.97	0.92	1.13	0.45	0.62	0.77	8.7	13.69	6.68
7	0.97	1.23	0.82	0.54	1.1	0.53	0.92	1.2	0.89	0.64	0.69	0.76	8.8	14.27	6.64
8	1.1	0.54	0.65	0.57	0.69	0.51	0.96	1.68	0.74	0.63	0.44	0.77	8.69	13.59	6.02
9	1.06	0.91	0.73	0.58	0.78	0.49	0.93	1.3	0.91	–	–	–	9.01	14.27	6.54
10	1.01	0.33	0.77	0.53	0.76	0.46	0.93	0.89	1.07	0.63	0.58	0.65	8.46	13.63	6.61
11	1.04	0.26	0.74	0.54	0.82	0.46	1.03	1.33	0.91	–	–	–	8.64	14.24	0.66
12	–	–	–	–	–	–	1.04	1.51	0.55	0.7	0.7	0.8	8.88	14	0.66
13	–	–	–	–	–	–	1.03	1.41	0.75	0.79	0.58	0.82	8.82	14.13	6.55
14	–	–	–	–	–	–	0.86	1.19	0.86	0.51	0.64	0.74	8.63	14	6.62
15	1.07	0.82	0.79	0.54	0.67	0.48	1.05	1.45	0.75	0.69	0.63	0.75	8.96	14.75	6.84
16	1.01	0.27	0.85	0.53	0.89	0.5	1.01	1.12	0.86	0.71	0.64	0.78	8.94	14.59	6.8
17	0.98	0.95	0.77	0.53	0.75	0.47	1.01	1.51	0.81	0.48	0.71	0.87	8.93	14.17	6.8
18	1.15	0.77	0.9	0.67	0.66	0.59	1	1.51	0.8	0.52	0.65	0.59	6.4	10.77	5.37
19	1.15	0.96	0.86	0.68	0.9	0.55	1.04	1.6	0.9	0.55	0.46	0.73	7.14	14.35	5.85
20	1.05	0.64	0.76	0.53	0.83	0.46	1	1.2	0.67	0.59	0.85	0.94	8.23	14.09	6.3
21	0.97	0.88	0.96	0.65	0.91	0.67	0.78	0.83	0.5	0.64	0.33	0.89	9.52	15.15	7.6
22	1.01	1.11	0.58	0.6	0.87	0.45	1.01	1.16	0.58	0.91	0.63	0.85	8.84	14.05	6.4
23	1.01	0.57	0.78	0.55	0.88	0.45	1.01	1.38	0.85	0.57	0.69	0.74	8.97	13.9	6.72
24	0.79	0.76	1.08	0.79	0.59	0.66	0.81	0.79	1.66	0.53	0.66	0.62	5.82	8.22	4.69
25	1.15	0.58	0.87	0.61	0.75	0.61	0.89	1.27	0.4	0.33	0.39	0.73	8.73	13.73	6.44
26	0.93	0.48	0.94	0.57	0.75	0.51	0.91	1.07	0.68	0.61	0.36	0.92	8.9	14.07	6.84

The indices of the techniques are listed in Table 3

Table 16 Separate evaluation of techniques for Convolutional Neural Network, BERT, and RoBERTa (RB), standard deviation across the folds (%)

Model	hb hate			hb off			hasoc			olid			heval		
	CNN	BERT	RB	CNN	BERT	RB	CNN	BERT	RB	CNN	BERT	RB	CNN	BERT	RB
baseline	1.16	1.13	0.79	0.44	0.36	0.45	0.22	0.53	0.7	0.87	0.42	0.67	5.97	6.19	7.27
5	0.82	0.77	1.04	0.56	0.56	0.54	0.19	0.54	0.72	1.1	0.68	0.29	6.41	6.67	7.12
6	0.82	0.69	0.92	0.57	0.37	0.41	0.21	0.61	0.67	1.1	0.14	0.35	6.41	0.94	7.34
7	1.12	0.55	0.62	0.65	0.49	0.51	0.18	0.42	0.54	1.2	0.34	0.93	6.71	6.45	7.39
8	1.51	1.19	0.78	0.67	0.46	0.6	0.31	0.11	0.84	1.49	0.23	0.44	5.62	6.51	7.43
9	1.05	0.76	1.45	0.73	0.51	0.41	0.21	0.67	0.86	–	–	–	4.22	6.47	7.28
10	0.73	0.68	0.58	0.32	0.53	0.63	0.12	0.66	0.12	0.88	0.64	0.47	5.81	6.73	7.33
11	1.02	1.12	1.02	0.54	0.38	0.57	0.18	0.46	0.28	–	–	–	4.96	6.25	7.39
12	–	–	–	–	–	–	0.23	0.14	1.07	0.26	0.29	0.61	5.38	6.37	7.57
13	–	–	–	–	–	–	0.57	0.26	0.23	2.53	0.4	0.58	5.42	6.01	6.17
14	–	–	–	–	–	–	0.36	0.43	0.34	0.53	0.24	0.37	6.59	5.55	6.76
15	0.73	1.06	0.83	0.55	0.42	0.42	0.53	0.58	0.2	0.85	0.37	0.31	5.63	6.43	6.88
16	0.68	0.7	0.86	0.59	0.35	0.47	0.35	0.64	0.51	0.8	0.97	1.24	6.4	6.45	6.38
17	1.08	0.48	0.97	0.48	0.52	0.41	0.27	0.67	0.38	0.54	0.14	0.59	6.34	6.62	6.07
18	0.6	0.57	0.64	0.8	0.5	0.57	0.35	0.35	0.22	0.43	0.53	0.25	4.47	4.82	5.75
19	1.21	0.51	1.13	0.67	0.47	0.55	0.22	0.16	0.17	0.61	0.29	0.6	4.24	6.38	4.43
20	0.84	1.1	0.63	0.49	0.28	0.61	0.23	0.57	0.61	1.36	0.91	2.76	6.84	6.82	7.53
21	1.26	0.86	1.01	0.97	0.31	0.64	1.27	0.76	0.06	0.48	0.63	0.53	5.86	7.53	7.83
22	1.03	1.14	0.96	0.56	0.34	0.48	0.43	0.58	0.59	0.55	0.33	0.68	6.37	6.73	7.19
23	1.06	1	0.8	0.44	0.32	0.42	0.21	0.34	0.7	0.52	0.45	0.51	6.04	7.01	8
24	1.69	0.78	0.58	0.64	0.34	0.55	0.53	0.2	0.83	0.14	0.27	0.72	6.52	6.14	6.41
25	0.79	1.06	0.38	0.63	0.31	0.55	0.36	0.61	0.09	0.41	0.41	0.26	5.85	6.71	9.31
26	0.93	0.95	0.92	0.65	0.44	0.57	0.31	0.94	0.47	1.22	0.39	0.51	6.43	6.64	7.25

The indices of the techniques are listed in Table 3

References

Alam S, Yao N (2019) The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis. Comput Math Org Theory 25:319–335

Alfina I, Mulia R, Fanany MI, Ekanata Y (2017) Hate speech detection in the Indonesian language: a dataset and preliminary study. In: 2017 international conference on advanced computer science and information systems (ICACSIS). IEEE, pp 233–238

Alonso P, Saini R, Kovacs G (2020) TheNorth at SemEval-2020 task 12: hate speech detection using Roberta. In: Proceedings of the fourteenth workshop on semantic evaluation, pp 2197–2202

Alrehili A (2019) Automatic hate speech detection on social media: a brief survey. In: 2019 IEEE/ACS 16th international conference on computer systems and applications (AICCSA). IEEE, pp 1–6

Alshalan R, Al-Khalifa H (2020) A deep learning approach for automatic hate speech detection in the Saudi Twittersphere. Appl Sci 10(23):8614

Ameer I, Siddiqui MHF, Sidorov G, Gelbukh A (2019) CIC at SemEval-2019 task 5: simple yet very efficient approach to hate speech detection, aggressive behavior detection, and target classification in Twitter. In: Proceedings of the 13th international workshop on semantic evaluation, pp 382–386

Angiani G, Ferrari L, Fontanini T, Fornacciari P, Iotti E, Magliani F, Manicardi S (2016) A comparison between preprocessing techniques for sentiment analysis in twitter. In: KDWeb

Ashraf N, Rafiq A, Butt S, Shehzad HMF, Sidorov G, Gelbukh AF (2022) Youtube based religious hate speech and extremism detection dataset with machine learning baselines. J Intell Fuzzy Syst 42:4769–4777

Badjatiya P, Gupta S, Gupta M, Varma V (2017) Deep learning for hate speech detection in tweets. In: Proceedings of the 26th international conference on world wide web companion, pp 759–760

Bai Q, Dan Q, Mu Z, Yang M (2019) A systematic review of emoji: current research and future perspectives. Front Psychol 10:2221

Balouchzahi F, Shashirekha H (2020) Las for hasoc-learning approaches for hate speech and offensive content identification. In: FIRE (working notes), pp 145–151

Banerjee S, Sarkar M, Agrawal N, Saha P, Das M (2021) Exploring transformer based models to identify hate speech and offensive content in English and Indo-Aryan languages. arXiv preprint arXiv:2111.13974

Barbieri F, Camacho-Collados J, Espinosa Anke L, Neves L (2020) TweetEval: unified benchmark and comparative evaluation for tweet classification. In: Findings of the association for computational linguistics: EMNLP 2020, pp. 1644–1650. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.148 . https://aclanthology.org/2020.findings-emnlp.148

Baruah A, Barbhuiya F, Dey K (2019) ABARUAH at SemEval-2019 task 5: bi-directional LSTM for hate speech detection. In: Proceedings of the 13th international workshop on semantic evaluation, pp 371–376

Basile V, Bosco C, Fersini E, Debora N, Patti V, Pardo FMR, Rosso P, Sanguinetti M (2019) SemEval-2019 task 5: multilingual detection of hate speech against immigrants and women in Twitter. In: 13th international workshop on semantic evaluation. Association for Computational Linguistics, pp 54–63

Bhandari A, Shah SB, Thapa S, Naseem U, Nasim M (2023) CrisisHateMM: multimodal analysis of directed and undirected hate speech in text-embedded images from Russia-Ukraine conflict. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR) workshops, pp 1993–2002

Bird S (2006) NLTK: the natural language toolkit. In: Proceedings of the COLING/ACL 2006 interactive presentation sessions, pp 69–72

Bölücü N, Canbay P (2021) Hate speech and offensive content identification with graph convolutional networks. In: Forum for information retrieval evaluation (working notes)(FIRE), CEUR-WS.org, pp 44–51

Caselli T, Basile V, Mitrović J, Granitzer M (2021) HateBERT: retraining BERT for abusive language detection in English. In: Proceedings of the 5th workshop on online abuse and harms (WOAH 2021), pp 17–25

Caselli T, Basile V, Mitrović J, Kartoziya I, Granitzer M (2020) I feel offended, don't be abusive! implicit/explicit messages in offensive and abusive language. In: Proceedings of the 12th language resources and evaluation conference, pp 6193–6202

Chollet F et al. Keras. https://github.com/fchollet/keras

Conneau A, Khandelwal K, Goyal N, Chaudhary V, Wenzek G, Guzmán F, Grave É, Ott M, Zettlemoyer L, Stoyanov V (2020) Unsupervised cross-lingual representation learning at scale. In: Proceedings of the 58th annual meeting of the association for computational linguistics, pp 8440–8451

Das AK, Al Asif A, Paul A, Hossain MN (2021) Bangla hate speech detection on social media using attention-based recurrent neural network. J Intell Syst 30(1):578–591

Davidson T, Bhattacharya D, Weber I (2019) Racial bias in hate speech and abusive language detection datasets. In: Proceedings of the third workshop on abusive language online, pp 25–35

Davidson T, Warmsley D, Macy M, Weber I (2017) Automated hate speech detection and the problem of offensive language. In: Proceedings of the international AAAI conference on web and social media, vol 11, pp 512–515

Devlin J, Chang M-W, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the North American chapter of the Association for Computational Linguistics: human language technologies, volume 1 (long and short papers). Association for Computational Linguistics, Minneapolis, Minnesota, pp 4171–4186

Do HT-T, Huynh HD, Van Nguyen K, Nguyen NL-T, Nguyen AG-T (2019) Hate speech detection on Vietnamese social media text using the bidirectional-LSTM model. In: The sixth international workshop on Vietnamese language and speech processing VLSP 2019

Dogru HB, Tilki S, Jamil A, Hameed AA (2021) Deep learning-based classification of news texts using Doc2vec model. In: 2021 1st international conference on artificial intelligence and data analytics (CAIDA). IEEE, pp 91–96

Fersini E, Nozza D, Rosso P (2018) Overview of the Evalita 2018 task on automatic misogyny identification (AMI). In: CEUR workshop proceedings. CEUR-WS, vol 2263, pp 1–9

Fersini E, Rosso P, Anzovino M (2018) Overview of the task on automatic misogyny identification at IberEval 2018. In: CEUR workshop proceedings. CEUR-WS, vol 2150, pp 214–228

Fortuna P, Nunes S (2018) A survey on automatic detection of hate speech in text. ACM Comput Surv CSUR 51(4):1–30

Fromknecht J, Palmer A (2020) UNT linguistics at SemEval-2020 task 12: linear SVC with pre-trained word embeddings as document vectors and targeted linguistic features. In: Proceedings of the fourteenth workshop on semantic evaluation, pp 2209–2215

Garain A, Basu A (2019) The titans at SemEval-2019 task 5: detection of hate speech against immigrants and women in Twitter. In: Proceedings of the 13th international workshop on semantic evaluation, pp 494–497

Garouani M, Chrita H, Kharroubi J (2021) Sentiment analysis of Moroccan tweets using text mining. In: Digital technologies and applications: proceedings of ICDTA 21, Fez, Morocco. Springer, pp 597–608

Glazkova A, Kadantsev M, Glazkov M (2021) Fine-tuning of pre-trained transformers for hate, offensive, and profane content

detection in English and Marathi. In: FIRE 2021 working notes, pp 52–62
Guibon G, Ochs M, Bellot P (2016) From emojis to sentiment analysis. In: WACAI 2016
Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
Huang X, Xing L, Dernoncourt F, Paul MJ (2020) Multilingual twitter corpus and baselines for evaluating demographic bias in hate speech recognition. In: LREC
Hu R, Dorris W, Vishwamitra N, Luo F, Costello M (2020) On the impact of word representation in hate speech and offensive language detection and explanation. In: Proceedings of the tenth ACM conference on data and application security and privacy, pp 171–173
Jianqiang Z, Xiaolin G (2017) Comparison research on text preprocessing methods on twitter sentiment analysis. IEEE Access 5:2870–2879
Joulin A, Grave E, Bojanowski P, Mikolov T (2016) Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759
Kadhim AI (2018) An evaluation of preprocessing techniques for text classification. Int J Comput Sci Inf Secur IJCSIS 16(6):22–32
Kaibi I, Satori H (2019) A comparative evaluation of word embeddings techniques for twitter sentiment analysis. In: 2019 international conference on wireless technologies, embedded and intelligent systems (WITS). IEEE, pp 1–4
Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). Association for Computational Linguistics, Doha, Qatar, pp 1746–1751
Kirk H, Yin W, Vidgen B, Röttger P (2023) SemEval-2023 task 10: explainable detection of online sexism. In: Proceedings of the 17th international workshop on semantic evaluation (SemEval-2023). Association for Computational Linguistics, Toronto, Canada, pp 2193–2210. https://aclanthology.org/2023.semeval-1.305
Kodali P, Bhatnagar A, Ahuja N, Shrivastava M, Kumaraguru P (2022) HashSet—a dataset for hashtag segmentation. arXiv preprint arXiv:2201.06741
Krouska A, Troussas C, Virvou M (2016) The effect of preprocessing techniques on twitter sentiment analysis. In: 2016 7th international conference on information, intelligence, systems & applications (IISA). IEEE, pp 1–5
Liao W, Zeng B, Liu J, Wei P, Cheng X, Zhang W (2021) Multi-level graph neural network for text sentiment analysis. Comput Electr Eng 92:107096
Li M, Liao S, Okpala E, Tong M, Costello M, Cheng L, Hu H, Luo F (2021) COVID-hateBERT: a pre-trained language model for COVID-19 related hate speech detection. In: 2021 20th IEEE international conference on machine learning and applications (ICMLA), pp 233–238. IEEE
Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V (2019) Roberta: a robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692
Loshchilov I, Hutter F (2018) Decoupled weight decay regularization. In: International conference on learning representations
Luu ST, Nguyen HP, Van Nguyen K, Nguyen NL-T (2020) Comparison between traditional machine learning models and neural network models for Vietnamese hate speech detection. In: 2020 RIVF international conference on computing and communication technologies (RIVF). IEEE, pp 1–6
MacAvaney S, Yao H-R, Yang E, Russell K, Goharian N, Frieder O (2019) Hate speech detection: challenges and solutions. PLoS ONE 14(8):0221152
Mandl T, Modha S, Kumar M A, Chakravarthi BR (2020) Overview of the HASOC track at fire 2020: hate speech and offensive language identification in Tamil, Malayalam, Hindi, English and German. In: Forum for information retrieval evaluation, pp 29–32
Mandl T, Modha S, Majumder P, Patel D, Dave M, Mandlia C, Patel A (2019) Overview of the HASOC track at fire 2019: hate speech and offensive content identification in Indo-European languages. In: Proceedings of the 11th forum for information retrieval evaluation, pp 14–17
Menini S, Aprosio AP, Tonelli S (2021) Abuse is contextual, what about NLP? The role of context in abusive language annotation and detection. arXiv preprint arXiv:2103.14916
Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781
Mishra AK, Saumya S, Kumar A (2020) Iiit_dwd@hasoc 2020: identifying offensive content in Indo-European languages. In: FIRE (working notes), pp 139–144
Modha S, Mandl T, Majumder P, Satapara S, Patel T, Madhu H (2022) Overview of the HASOC subtrack at fire 2022: identification of conversational hate-speech in Hindi-English code-mixed and German language. Working notes of FIRE
Modha S, Mandl T, Shahi GK, Madhu H, Satapara S, Ranasinghe T, Zampieri M (2021) Overview of the HASOC subtrack at fire 2021: hate speech and offensive content identification in English and Indo-Aryan languages and conversational hate speech. In: Forum for information retrieval evaluation, pp 1–3
Mohammad F (2018) Is preprocessing of text really worth your time for toxic comment classification? In: Proceedings on the international conference on artificial intelligence (ICAI). The Steering Committee of The World Congress in Computer Science, Computer, pp 447–453
Montejo-Ráez A, Jiménez-Zafra SM, Garcia-Cumbreras MA, Díaz-Galiano MC (2019) SINAI-DL at SemEval-2019 task 5: recurrent networks and data augmentation by paraphrasing. In: Proceedings of the 13th international workshop on semantic evaluation, pp 480–483
Naseem U, Razzak I, Hameed IA (2019) Deep context-aware embedding for abusive and hate speech detection on twitter. Aust J Intell Inf Process Syst 15(3):69–76
Naseem U, Razzak I, Eklund PW (2021) A survey of pre-processing techniques to improve short-text quality: a case study on hate speech detection on twitter. Multimedia Tools Appl 80(28):35239–35266
Nobata C, Tetreault J, Thomas A, Mehdad Y, Chang Y (2016) Abusive language detection in online user content. In: Proceedings of the 25th international conference on world wide web, pp 145–153
Nugroho K, Noersasongko E, Fanani AZ, Basuki RS (2019) Improving random forest method to detect hatespeech and offensive word. In: 2019 international conference on information and communications technology (ICOIACT). IEEE, pp 514–518
Oliveira DN, Merschmann LHDC (2021) Joint evaluation of preprocessing tasks with classifiers for sentiment analysis in Brazilian Portuguese language. Multimedia Tools Appl 80:15391–15412
Oriola O, Kotzé E (2020) Evaluating machine learning techniques for detecting offensive and hate speech in South African tweets. IEEE Access 8:21496–21509. https://doi.org/10.1109/ACCESS.2020.2968173
Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, et al (2019) Pytorch: an imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 32
Pavlopoulos J, Sorensen J, Laugier L, Androutsopoulos I (2021) SemEval-2021 task 5: toxic spans detection. In: Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021), pp 59–69
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830

Pennington J, Socher R, Manning C.D (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543

Plaza-Del-Arco FM, Molina-González MD, Ureña-López LA, Martín-Valdivia MT (2021) A multi-task learning approach to hate speech detection leveraging sentiment analysis. IEEE Access 9:112478–112489

Poletto F, Basile V, Sanguinetti M, Bosco C, Patti V (2021) Resources and benchmark corpora for hate speech detection: a systematic review. Lang Resour Eval 55(2):477–523

Porter MF (2001) Snowball: a language for stemming algorithms

Rajapakse TC (2019) Simple transformers. https://github.com/ThilinaRajapakse/simpletransformers

Ramachandran D, Parvathi R (2019) Analysis of twitter specific preprocessing technique for tweets. Procedia Comput Sci 165:245–251

Ranasinghe T, Hettiarachchi H (2020) Brums at SemEval-2020 task 12: transformer based multilingual offensive language identification in social media. In: Proceedings of the fourteenth workshop on semantic evaluation, pp 1906–1915

Renault T (2020) Sentiment analysis and machine learning in finance: a comparison of methods and models on one million messages. Digit Finance 2(1–2):1–13

Reuter J, Pereira-Martins J, Kalita J (2016) Segmenting Twitter hashtags. Int J Nat Lang Comput 5(4):23–36

Rogers A, Kovaleva O, Rumshisky A (2020) A primer in BERTology: what we know about how BERT works. Trans Assoc Comput Ling 8:842–866

Saeed AM, Ismael AN, Rasul DL, Majeed RS, Rashid TA (2022) Hate speech detection in social media for the Kurdish language. In: Proceedings of the ICR'22 international conference on innovations in computing research. Springer, pp 253–260

Saeed NM, Helal NA, Badr NL, Gharib TF (2018) The impact of spam reviews on feature-based sentiment analysis. In: 2018 13th international conference on computer engineering and systems (ICCES). IEEE, pp 633–639

Saeed NM, Helal NA, Badr NL, Gharib TF (2020) An enhanced feature-based sentiment analysis approach. Wiley Interdiscip Rev Data Min Knowl Discov 10(2):1347

Saeed RM, Rady S, Gharib TF (2021) Optimizing sentiment classification for Arabic opinion texts. Cogn Comput 13(1):164–178

Saeed RM, Rady S, Gharib TF (2022) An ensemble approach for spam detection in Arabic opinion texts. J King Saud Univ Comput Inf Sci 34(1):1407–1416

Schmidt A, Wiegand M (2019) A survey on hate speech detection using natural language processing. In: Proceedings of the fifth international workshop on natural language processing for social media, April 3, 2017, Valencia, Spain. Association for Computational Linguistics, pp 1–10

Silva SC, Ferreira TC, Ramos RMS, Paraboni I (2020) Data driven and psycholinguistics motivated approaches to hate speech detection. Computación y Sistemas 24

Štrimaitis R, Stefanovič P, Ramanauskaitė S, Slotkienė A (2021) Financial context news sentiment analysis for the Lithuanian language. Appl Sci 11(10):4443

Symeonidis S, Effrosynidis D, Arampatzis A (2018) A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis. Expert Syst Appl 110:298–310

Thapa S, Jafri FA, Hürriyetoğlu A, Vargas F, Lee RK-W, Naseem U (2023) Multimodal hate speech event detection—shared task 4, CASE 2023. In: Proceedings of the 6th workshop on challenges and applications of automated extraction of socio-political events from text (CASE)

Toraman C, Şahinuç F, Yılmaz EH (2022) Large-scale hate speech detection with cross-domain transfer. arXiv preprint arXiv:2203.01111

Wallace E, Wang Y, Li S, Singh S, Gardner M (2019) Do NLP models know numbers? Probing numeracy in embeddings. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp 5307–5315

Wang B, Ding Y, Liu S, Zhou X (2019) Ynu_wb at HASOC 2019: ordered neurons LSTM with attention for identifying hate speech and offensive language. In: FIRE (working notes), pp 191–198

Wang S, Liu J, Ouyang X, Sun Y (2020) Galileo at SemEval-2020 task 12: multi-lingual learning for offensive language identification using pre-trained language models. In: Proceedings of the fourteenth workshop on semantic evaluation, pp 1448–1455

Wang D, Liu P, Zheng Y, Qiu X, Huang X-J (2020) Heterogeneous graph neural networks for extractive document summarization. In: Proceedings of the 58th annual meeting of the association for computational linguistics, pp 6209–6219

Wiedemann G, Yimam SM, Biemann C (2020) UHH-LT at SemEval-2020 task 12: fine-tuning of pre-trained transformer networks for offensive language detection. arXiv preprint arXiv:2004.11493

Wiegand M, Ruppenhofer J, Kleinbauer T (2019) Detection of abusive language: the problem of biased datasets. In: Proceedings of the 2019 conference of the North American chapter of the Association for Computational Linguistics: human language technologies, volume 1 (long and short papers), pp 602–608

Yin W, Zubiaga A (2021) Towards generalisable hate speech detection: a review on obstacles and solutions. PeerJ Comput Sci 7:598

Zampieri M, Malmasi S, Nakov P, Rosenthal S, Farra N, Kumar R (2019) SemEval-2019 task 6: identifying and categorizing offensive language in social media (offenseval). In: Proceedings of the 13th international workshop on semantic evaluation, pp 75–86

Zampieri M, Nakov P, Rosenthal S, Atanasova P, Karadzhov G, Mubarak H, Derczynski L, Pitenis Z, Çöltekin Ç (2020) SemEval-2020 task 12: multilingual offensive language identification in social media (offenseval 2020). In: Proceedings of the fourteenth workshop on semantic evaluation, pp 1425–1447

Zhang Z, Han X, Liu Z, Jiang X, Sun M, Liu Q (2019) ERNIE: enhanced language representation with informative entities. In: Proceedings of the 57th annual meeting of the association for computational linguistics, pp 1441–1451

Zhou Y, Yang Y, Liu H, Liu X, Savage N (2020) Deep learning based fusion approach for hate speech detection. IEEE Access 8:128923–128929

Zhou X, Yong Y, Fan X, Ren G, Song Y, Diao Y, Yang L, Lin H (2021) Hate speech detection based on sentiment knowledge sharing. In: Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers), pp 7158–7166