UNIVERSITÄT PASSAU

HATE SPEECH DETECTION

Using Hurtlex Lexicon

Team 10
Miran M. Rashid Salim Fares

supervisor: Dr. Jelena Mitrovic

July 22, 2020

Computer Science and Mathematics

Contents

List of Figures

List of Tables

1 Introduction

In 2010, fewer than 1 billion people were signed on. That number has since tripled according to USA Today Network. Social media, giving the impression of freedom of speech, allow people to share their opinions and feelings in immense speed and reach. This also brings negative consequences, as social media has become a powerful tool for spreading hateful opinions, often reflecting social and political tensions [1]. Many users are taking advantage of social media to express their hatred and hostility, with a possibility to influence other people, too. In addition, there is a strong connection between hate speech and hate crime [10], therefore it is strongly required to develop effective methods for automatic detection of hate speech. Popular social networks, like Twitter and Facebook, are facing criticism for not doing enough to prevent hate speech [3].

The definitions of **Hate Speech** (HS) slightly vary [1, 3, 4, 10], but they unite in one point - hate speech is any communication that targets disadvantaged person, social group or minority in a way that is potentially harmful to the object or directly disparages it. This speech attack is typically based on characteristics such as gender, race, colour, ethnicity, sexual orientation, nationality, social status etc. More generally, HS can be defined as any type of communication which is abusive, insulting, humiliating, intimidating, harassing or leads to violence or discrimination.

Although offensive language is very often a part of hate speech, the presence of an offensive language in a speech does not necessarily imply that this speech is a HS, which makes the auto-detection pretty complicated [3]. Nevertheless, more lexicons of offensive language and other expressions, that are not offensive themselves but still have an offensive potential, have been created and shown useful, e.g. HurtLex [1] or Hatebase.org. Using these lexicons, we can effectively detect potentially offensive speech or even HS [3].

1.1 Hurtlex

HurtLex is a multilingual lexicon of categorised hate words (or words to hurt), that was created semi-automatically from an Italian hate lexicon[1]. HurtLex was created by extracting the lemmas from this lexicon and after that, with the help of MultiWordNet and BabelNet dictionaries, translated into over 50 other languages. Thanks to the dictionaries many of the lemmas maintained their semantic definition even after the translations, though many mistakes can only be fixed manually. There are 17 different HS categories in HurtLex (table 1) which can be used for detecting different kinds of hate speech. Every specific HS attack, such as misogyny, is mostly employing just a few chosen categories of HurtLex. This gives a potential for varied applications of HurtLex.

[1] This lexicon was developed by Tullio De Mauro.

Label	Description
PS	negative stereotypes ethnic slurs
RCI	locations and demonyms
PA	professions and occupations
DDF	physical disabilities and diversity
DDP	cognitive disabilities and diversity
DMC	moral and behavioral defects
IS	words related to social and economic disadvantage
OR	plants
AN	animals
ASM	male genitalia
ASF	female genitalia
PR	words related to prostitution
OM	words related to homosexuality
QAS	with potential negative connotations
CDS	derogatory words
RE	felonies and words related to crime and immoral behavior
SVP	words related to the seven deadly sins of the Christian tradition

Table 1: Hurtlex 17 HS categories

2 Methodology

To build an intelligent approach to Detection of hate speech and offensive language, the methodology designed(figure1). The first phase is modifying Hurtlex then collecting and Pre-processing the datasets. Secondly Transforms texts (tweets) into features That the machine learning algorithms are understandable. Thirdly, design the model of smart predictions. Ultimately, is evaluation Of the model in date.

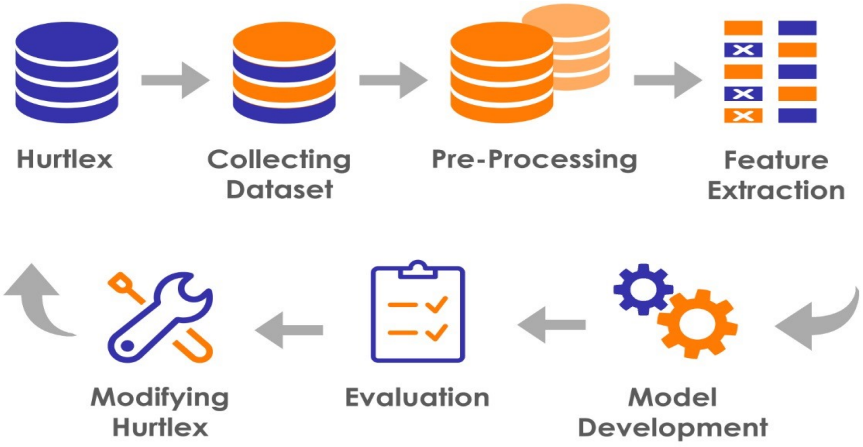


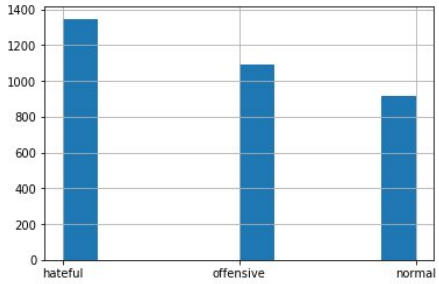
Figure 1: Methodology

2.1 Modifying Hurtlex

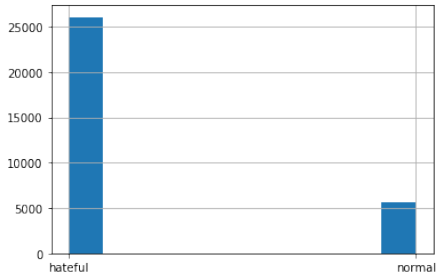
For this project, we are just focusing on the Arabic and English versions. The Arabic lexicon consists of 2367 Arabic hate words along with different semantic categories of hate. Since the Arabic version was machine-translated from the Italian version they had to be viewed and edited. We modified 120 words deleted 1181 words and added 105 words. The English lexicon provides 8229 English lemmas related to hate speech as well, but due to machine-translated from The Italian version, it also needs modification, despite we are not native English speakers, we could delete 359 lemmas and add 65 lemmas. After training our models we will extract more words and expressions from the data-set.

2.2 Data

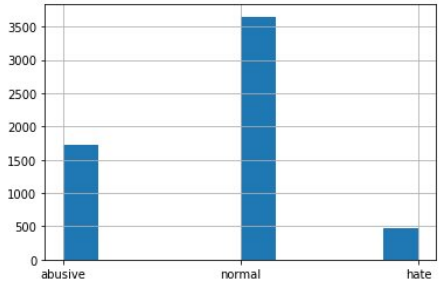
We used four datasets for our project. AR dataset [9], L-HSAB [8] and algazeera.net [7] for the Arabic part which gave us in total 41991 tweets and comments which we treated like dependent tweets. We unified the labels and reduced the to three labels only hateful, offensive and normal. AR-dataset (figure 2a) contains tweets related to gender, sexual orientation, religion, disability. L-HSAB (figure 2c) combines Syrian and Lebanese political tweets. Aljazeera.net (figure 2b) contains news comments on Aljazeera.net which means the majority of it is also political. En-dataset [3] which consists 24748 tweets that contain content that is racist, sexist, homophobic, and offensive in many other ways, as shown in (figure 2d) each tweet labeled as one of three categories: hate speech, offensive, or neither offensive nor hate speech.



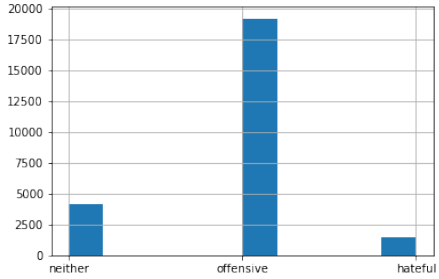
(a) Ar-dataset



(b) Aljazeera.net



(c) L-HSAB dataset



(d) En-dataset

Figure 2: (a)Ar-dataset, (b)comments on Aljazeera.net, (c)L-HSAB dataset and (d) En-dataset

2.3 Data Pre-Processing

The data pre-processing starts by cleaning the data from the redundancy of tweets and the character that appears more than once,unifying multiple versions of some characters into just one, for instance

- ١ ~ اااإأٔ

To be more accurate and readable all numbers, symbols, punctuation, hashtags, web addresses, (for Arabic all non-Arabic characters) and emoji has been removed from the data and Stop words were filtered as well. The normalization for Arabic tweets can be done by replacing the Arabic characters from the standard writing way by the colloquial-writing way. Tweets tokenization is treated with dividing the tweets into a set of words based on the white-space delimited and performed a stemming on the tweets.

2.4 Feature Extraction

Several treatments have recently been proposed To represent texts in such a way as to be understandable by machine algorithms. In order to approach text representation, the different ideas in this project have implemented to obtain relevant features from the datasets as the following which we will brief them later :

- Count Vectors.
- NLP based.

- TF-IDF Vectors.
- Word Embedding.

2.4.1 Count Vectors as features

Count Vector [2]is a matrix notation of the dataset in which every row represents a tweet from the corpus, every column represents a term from the corpus, and every cell represents the frequency count of a particular term in a particular tweet.

2.4.2 TF-IDF Vectors as features

TF-IDF is a shortened form of Term Frequency-Inverse Document Frequency, and it is a very simple algorithm for translating text into a meaningful representation of numbers[5]. TF-IDF score represents the relative importance of a term in the tweet and the entire corpus. TF-IDF score is composed by two terms: the first computes the normalized TF, the second term IDF, computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

$$TF(t) = \frac{\text{Number of times term t appears in a tweet}}{\text{Total number of terms in the tweet}}$$

$$IDF(t) = log_e\big(\frac{\text{Total number of tweets}}{\text{Number of tweets with term t in it}}\big)$$

$TF - IDF$ score is then calculated by multiplying those two terms $TF(t) * IDF(t)$.

$TF-IDF$Vectors can be generated at different levels of input tokens (words, characters, n-grams)

- **Word Level TF-IDF :** Matrix representing tf-idf scores of every term in different tweets.
- **N-gram Level TF-IDF :** N-grams are the combination of N terms together. This Matrix representing tf-idf scores of N-grams.
- **Character Level TF-IDF :**Matrix representing tf-idf scores of character level n-grams in the corpus.

2.4.3 Text / NLP based features

A number of extra text based features can also be created which sometimes are helpful for improving text classification models. The following features can be used :

1. Word Count of the tweet : total number of words in the tweet.
2. Character Count of the tweet : total number of characters in the tweet.
3. Average Word Density of the tweet : average length of the words used in the tweet.
4. Punctuation Count in the tweet : total number of punctuation marks in the tweet.
5. Hate Word : a binary label to indicate whether a tweet contains a hate word or not.
6. Hate Words Count : Number of hate words in the tweet

2.4.4 Word Embedding (only for Arabic)

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers [6]. Word embeddings can be trained using the input corpus itself or can be generated using pre-trained word embeddings such as Glove, FastText, and Word2Vec, AraVec. For Arabic We tried both approaches .

2.5 Model

There are many different choices of machine learning models which can be used to train a final model. We implemented following different classifiers for this purpose:

- Naive Bayes Classifier.
- Linear Classifier : Logistic. Regression
- Support Vector Machine (SVM).
- LSTM Just for Word Embeddings (only for Arabic)

We trained each of the previous models with all the feature vectors we had extracted. And then to get the best result we combined all those feature vectors by training the model with a specific feature vector and use its predictions as features. We concatenate those predictions with the NLP based features and then re-train the model, we got this idea from stackexchange[2].We used 70% of the corpus for training and 30% for testing (Evaluation).

3 Result

3.1 Evaluation Measures

The used performance evaluation measures[4] are accuracy, precision, recall, and F1 measure as the following:

- Accuracy is the ratio of correctly classified Hate and Normal tweets over all the correct and the incorrect number of classified tweets:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

- Precision is the ratio of tweets that correctly identified as Hate over the total number of Hate tweets:

$$\frac{TP}{TP + FP}$$

- recall (known as sensitivity) quantifies how much the classifier can recognize Hate tweets

$$\frac{TP}{TP + FN}$$

- F1measure is a metric for indicating the balance between precision and recall measures:

$$2\,x\frac{PrecisionxRecall}{Precision + Recall}$$

[2] https://stats.stackexchange.com/questions/297529/combining-text-and-non-text-features-in-a-classification-model

3.2 Dataset Analysis Using HurtLex

We tried to classify our tweets just by a simple comparison between our dataset and the lexicon before and after modifying the lexicon. We also wanted to take advantage of the terms category specified in the lexicon to find the dominant type of hate-speech in our corpus using word frequencies for lexicon terms and accumulate the frequencies of terms belong to same category.As seen in the table 2, the results are somehow good and even better than some of the models we tested in our experiments. The results Before and After modifying the lexicon is very close this is because most of the modifications were deleting unrelated words which helped with improving the Precision and for our dataset even little improvements mean almost thousand tweets more correctly classified.

	Before	After
Accuracy	0.6670	0.6676
Precision	0.6986	0.7188
Recall	0.8831	0.8262
F1-Score	0.7801	0.7688
Most Frequent Words	أله ،أهل ،أقتل God, Parents, Killing	أقتل ،أيول،أرهابِ killing, Pi**, Terrorism
Dominant Hate-Type	CDS, RE, AN	RE, CDS, AN

Table 2: Dataset Analysis Using HurtLex

3.3 Evaluation the Model

So far we have explored the datasets, features, and models. Now in our journey with machine learning, to obtain which models have the best result, we have to evaluate the models that we used. the evaluation of the models can be done by using four metrics: Accuracy, Precision, Recall, and F1-Score for each type of feature vectors. for natural simplicity we denote Accuracy by **A**, Precision by **P**, Recall by **R** and F1-Score by **F1**.

3.3.1 Evaluating The Model For Arabic

As the table 3 showes that, the Naive Bayes Classifier scored the best **A** for Count Vectors, best **P** for Count Vectors, Word-Level TF-IDF, N-Gram TF-IDF, and Character-Level TF-IDF Vectors, best **R** for NLP based features and best F1-Score for NLP based features. Logistic Regression Classifier scored the best **R** for Word-Level TF-IDF, Character-Level TF-IDF Vectors, NLP based features and best **P** for NLP based features and best **F1** for Count Vectors. SVM Classifier scored the best **A** for N-Gram TF-IDF, best Recall for Count Vectors, Word-Level TF-IDF, N-Gram TF-IDF, and Character-Level TF-IDF Vectors, best **F1** for Word-Level TF-IDF, N-Gram TF-IDF, and Character-Level TF-IDF Vectors. For combined features Logistic Regression scored the best scores in all four metrics, our final model surpassed all previous models in this field for Arabic.

3.3.2 Evaluating The Model For English

For the English dataset, overall the results of the evaluation of the models are great as in table 4 illustrate the evaluation results that we obtained. For the Count vectors, Logistic regression takes the best place at **A, R, F1** , and the Naive Bayes Classifier has the best result at **P**, In terms of WordLevel TF-IDF, SVM gives the best scores for both **R, F1**, and for the **A, P** not surprisingly Logistic regression takes the first place again, for both N-Gram Vectors and CharLevel Vectors, the best score at P is given by Logistic regression, and Naive Bayes Classifier respectively, and SVM has the best at **A, R, F1** as well. For the NLP, Logistic regression

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
NB, Count Vectors	0.7417	0.6934	0.5288	0.5666
NB,WordLevel TF-IDF	0.7005	0.8206	0.3976	0.3853
NB,N-Gram Vectors	0.6931	0.8430	0.3862	0.3663
NB, CharLevel Vectors	0.6993	0.8026	0.4045	0.3994
NB,NLP Features	0.6577	0.3984	0.3595	0.3344
LR,Count Vectors	0.7314	0.6571	0.5916	0.6163
LR,WordLevel TF-IDF	0.7481	0.6882	0.5641	0.6010
LR,N-Gram Vectors	0.6995	0.6553	0.4382	0.4534
LR,CharLevel Vectors	0.7505	0.6862	0.5731	0.6085
LR, NLP Features	0.6671	0.4825	0.3418	0.2871
SVM,Count Vectors	0.7152	0.6264	0.5917	0.6055
SVM,WordLevel TF-IDF	0.7386	0.6673	0.5733	0.6051
SVM,N-Gram Vectors	0.7002	0.6502	0.4533	0.4761
SVM,CharLevel Vectors	0.7455	0.6804	0.5770	0.6107
SVM,NLP Features	0.6664	0.4321	0.3379	0.2766
NB, Combined Features	0.9166	0.8033	0.8451	0.8206
LR,Combined Features	0.9875	0.9809	0.9674	0.9740
SVM,Combined Features	0.9837	0.9767	0.9614	0.9689

Table 3: Models Evaluation For Arabic

achieves the best scores at all of **A, P, R, F1**. The last and the least the Combined Features pretty much the result of Logistic regression and SVM close to each other.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
NB, Count Vectors	0.8619	0.7721	0.5397	0.5682
NB,WordLevel TF-IDF	0.7946	0.5810	0.3757	0.3696
NB,N-Gram Vectors	0.7802	0.5679	0.3463	0.3171
NB, CharLevel Vectors	0.8070	0.8140	0.4090	0.4235
NB,NLP Features	0.7695	0.4274	0.3386	0.3063
LR,Count Vectors	0.8901	0.7333	0.6902	0.7042
LR,WordLevel TF-IDF	0.8913	0.7640	0.6661	0.6913
LR,N-Gram Vectors	0.7952	0.7228	0.3944	0.4050
LR,CharLevel Vectors	0.8921	0.7582	0.6648	0.6865
LR, NLP Features	0.7804	0.4556	0.3594	0.3435
SVM,Count Vectors	0.8841	0.7185	0.6890	0.7009
SVM,WordLevel TF-IDF	0.8907	0.7525	0.6775	0.6993
SVM,N-Gram Vectors	0.8023	0.6945	0.4238	0.4504
SVM,CharLevel Vectors	0.8940	0.7640	0.6706	0.6931
SVM,NLP Features	0.7758	0.4412	0.3429	0.3124
NB, Combined Features	0.8050	0.7872	0.7025	0.7290
LR,Combined Features	0.9875	0.9649	0.9529	0.9587
SVM,Combined Features	0.9872	0.9670	0.9512	0.9588

Table 4: Models Evaluation For English

4 CONCLUSION

The results showed that we need a better way to represent text as features. We may have achieved the best score in this field but it took us to train 15 models and to collect data from different sources to get those results. As for the HurtLex lexicon, it still needs more modifications. There are so many words to be added. The lexicon could be used for analyzing the dataset using statistical methods to get an idea about the topic or the type of hate-speech in the corpus, it is also a good method to set a baseline for our approach but using it as a feature for classification did not make much difference especially when we combined all feature vectors. Arabic has enormous vocabulary and it is hard to train a good Word Embedding model due to lack of data. Our Word Embedding model did not perform as well as expected and it consumed more time than it did to train the other 15 models which we used in our approach t, so we did not included it in our final model.

References

[1] Elisa Bassignana, Valerio Basile, and Viviana Patti. "Hurtlex: A multilingual lexicon of words to hurt". In: *5th Italian Conference on Computational Linguistics, CLiC-it 2018*. Vol. 2253. CEUR-WS. 2018, pp. 1–6.

[2] Jason Brownlee. *How to Encode Text Data for Machine Learning with scikit-learn*. 2019. URL: https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/.

[3] Thomas Davidson et al. "Automated hate speech detection and the problem of offensive language". In: *Proceedings of the 11th International AAAI Conference on Web and Social Media*. ICWSM '17. Montreal, Canada, 2017, pp. 512–515.

[4] Hossam Faris et al. "Hate Speech Detection using Word Embedding and Deep Learning in the Arabic Language Context". In: Jan. 2020, pp. 453–460. DOI: 10.5220/0008954004530460.

[5] Prabhav Jain Gurpreet Singh. *TF-IDF: Vector representation of Text*. 2019. URL: https://www.commonlounge.com/discussion/99e86c9c15bb4d23a30b111b23e7b7b1.

[6] Abu Bakr Mohammad, Kareem Eissa, and Samhaa El-Beltagy. "AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP". In: *Procedia Computer Science* 117 (Nov. 2017), pp. 256–265. DOI: 10.1016/j.procs.2017.10.117.

[7] Hamdy Mubarak, Kareem Darwish, and Walid Magdy. "Abusive language detection on Arabic social media". In: *Proceedings of the first workshop on abusive language online*. 2017, pp. 52–56.

[8] Hala Mulki et al. "L-HSAB: A Levantine Twitter Dataset for Hate Speech and Abusive Language". In: *Proceedings of the Third Workshop on Abusive Language Online*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 111–118. DOI: 10.18653/v1/W19-3512. URL: https://www.aclweb.org/anthology/W19-3512.

[9] Nedjma Ousidhoum et al. *Multilingual and Multi-Aspect Hate Speech Analysis*. 2019. arXiv: 1908.11049 [cs.CL].

[10] Zeerak Waseem and Dirk Hovy. "Hateful symbols or hateful people? predictive features for hate speech detection on twitter". In: *Proceedings of the NAACL student research workshop*. 2016, pp. 88–93.