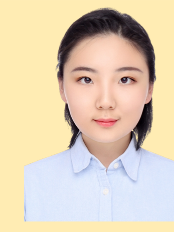
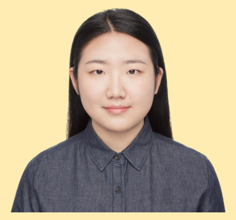
Emoji-Based Hate Speech Detection on Social Media

Supervisors
Gaël GUIBON
Christophe CERISARA


Nalin SRUN


Shuwei LYU


Shu ZHANG

UNIVERSITÉ DE LORRAINE
Institut des sciences du Digital Management & Cognition
COMPOSANTE DE L'UNIVERSITÉ DE LORRAINE
Loria
Laboratoire lorrain de recherche en informatique et ses applications

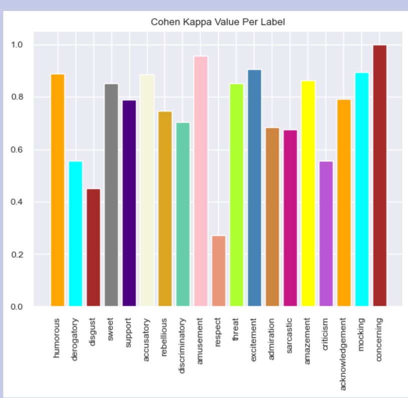
Introduction

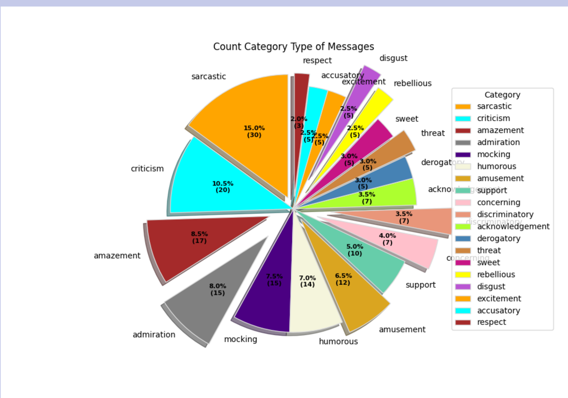
- The rise of Internet technology has led to an increase in harmful and abusive languages, including hate speech, in online user-generated content.
- Detecting and addressing online hate speech automatically and efficiently is essential for maintaining the well-being of the internet and individuals' psychological health.
- Emojis are increasingly used to convey hateful messages, posing a challenge for existing hate speech detection models focused on textual information.
- This research aims to enhance hate speech detection by developing a model specifically designed to detect emoji-based hate speech on social media.
- By utilizing the HATEMOJICHECK dataset and additional labeled data, the proposed model will be trained and evaluated to improve the detection of hateful content involving emojis.

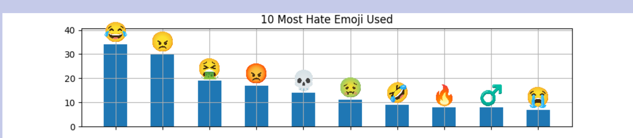
Dataset Annotation

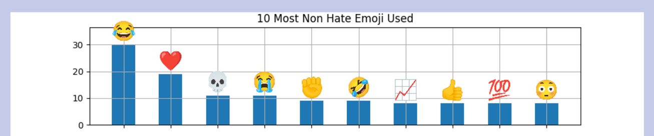
Annotated these sets by two annotators (Nalin Srun and Shu Zhang) before they were used to prepare a hate speech classifier dedicated to these particular emojis utilization. Moreover, the objective is not to consider all possible harmful emojis but as it were the ones from the sub-dataset.

Two annotators classified the dataset into positive and negative speech types, with some disagreements due to similar meanings and interpretation of emojis. The distribution of annotated message types varied, and the analysis of frequently used emojis showed differences between positive and negative speech. However, the small dataset size highlights the need for a larger dataset for more accurate results.









Dataset Creation

- Data from Kirk
- Data Scraped from Social Media (TikTok; Youtube)
- Illustration of the Scraped Dataset

Data Scraped from TikTok

- API called Tikhub (Version 3.15)

Data Scrpaed from Youtube

- Youtube-comment-downloader Library

Text Pre-processing

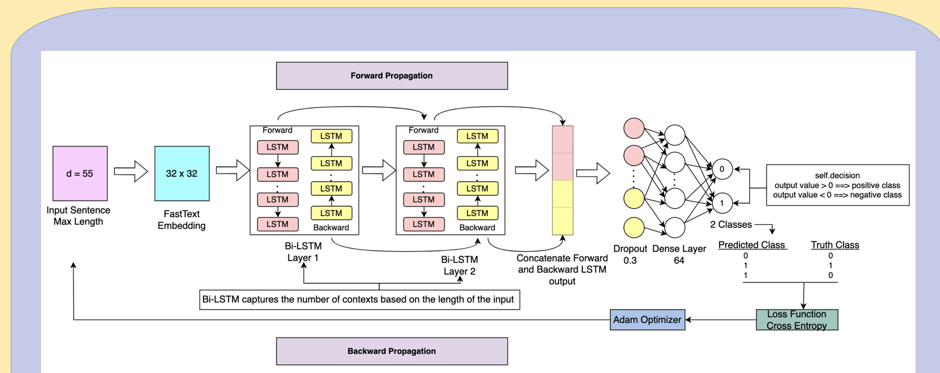
- Lowercasing the letters is performed to normalize the text and create a consistent vocabulary.
- Punctuations and emojis are retained in the pre-processing step, as they may carry important information.
- Different approaches for handling emojis are tested, including keeping the original emojis or substituting them with textual descriptions.
- Tokenization is performed using spaCy tokenizer, allowing successful separation of adjacent emojis.

Feature Extraction

- Three feature extraction methods are experimented with: Hashing Vectorizer, TF-IDF Vectorizer, and word vectors obtained from a self-trained FastText unsupervised model.
- Categorical encoding is applied to transform the target variables into a suitable format for the algorithms.

Proposed Machine Learning Models

- For binary classification, various models such as Logistic Regression, SGD Classifier, SVM, Decision Tree Classifier, Random Forest Classifier, LightGBM, BernoulliNB, MLP Classifier, and FastText (both self-trained and pre-trained) are employed.
- For multiclass classification, similar models are used for different sub-tasks, with an additional consideration for data imbalance and the use of data augmentation techniques like SMOTE.



Machine Learning

- Dataset Preparation
- Text Pre-processing
- Feature Extraction
- Proposed Machine Learning Models

Dataset Preparation

- The dataset from Kirk et al. (2022) consists of train, test, and validation sub-datasets, while an additional dataset scraped from social media is also available.
- For the binary classification task, the train and validation sub-datasets are combined as the train dataset, and two test datasets are used for evaluation.
- For the multiclass classification task, two sub-tasks are considered using different datasets for training and testing.

To be clearer, for each of the task, the train dataset and test dataset has been arranged as follows

Task		Train Dataset	Test Dataset
Binary Classification	1	Kirk's train+Kirk's validation	Kirk's test
	2	Kirk's train+Kirk's validation	scraped data
Multi-Classification	1	Kirk's train+Kirk's validation	Kirk's test
	2	80% of scraped data	20% of scraped data

Tokenized by NLTK

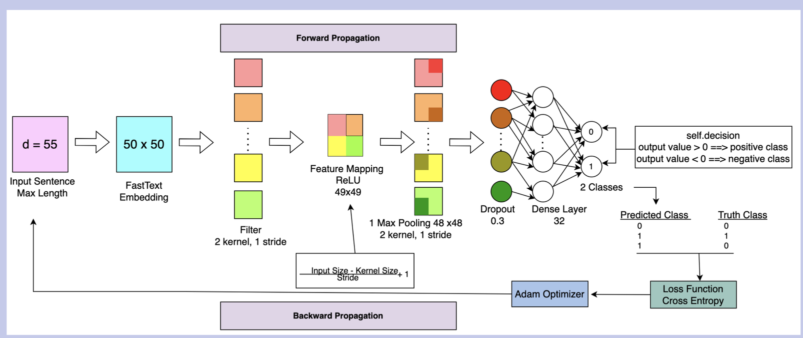
```
ex = 'i would love to 🔪🔪🔪 some 👳🏾'
print(preprocessing_text(ex))
[51] ✓ 0.0s
... ['i', 'would', 'love', 'to', '🔪🔪🔪', 'some', 'person wearing turban medium-dark skin tone']
```

Tokenized by spaCy

```
ex = 'i would love to 🔪🔪🔪 some 👳🏾'
print(preprocessing_text(ex))
[58] ✓ 0.0s
... ['i', 'would', 'love', 'to', 'kitchen knife', 'kitchen knife', 'kitchen knife', 'some', 'person wearing turban', 'medium-dark skin tone']
```

Feature Extraction

- Choose FastText for numerical representation
- Utilize pre-trained FastText embeddings



Deep Learning

- Data Preparation
- Feature Extraction
- Model Architecture of Binary Classification
- Model Architecture of Multiclass Classification

Dataset Preparation

Domain	Dataset	Sentences
Kirk	Train	4728
	Validation	591
	Test	593
TikTok	Test	100
YouTube	Test	100

Below describing the dataset we organize to build deep learning model

Phase 1

Domain	Dataset	Sentences
Kirk(Train+Validation)	Train	5319
TikTok+YouTube	Test	200

Phase 2

Domain	Dataset	Sentences
Kirk(Train+Validation+Test)	Train	5912
TikTok+YouTube	Test	200

Model Architecture for Binary Classification

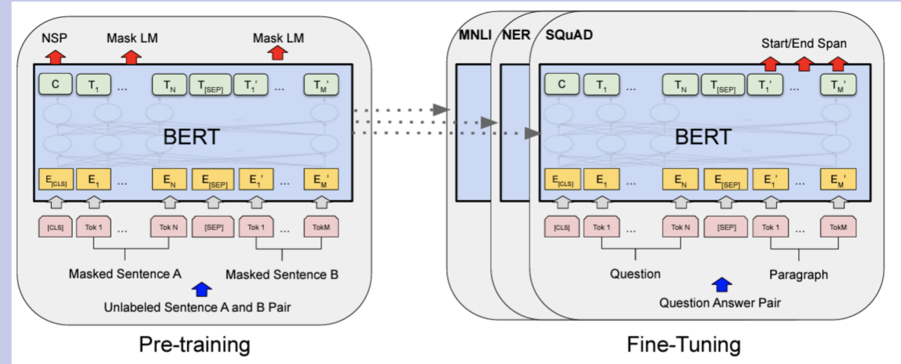
- Bi-LSTM
- CNN

Bi-LSTM	Value
Bi-LSTM Embedding Layer(1st training) FastText Embedding(2nd training)	300
Bi-LSTM Layer	256
Activation Function	NA
Dropout	0.3
Dense Layer	256
Loss Function	cross entropy
Optimizer	Adam

CNN	Value
CNN Embedding Layer(1st training) FastText Embedding(2nd training)	300
Convolutional Layer 1(Kernel=2,Stride=1,ReLU)	128
Convolutional Layer 2(Kernel=2,Stride=1,ReLU)	128
Convolutional Layer 3(Kernel=2,Stride=1,ReLU)	128
Pooling Layer	max_pool1d
Dropout	0.3
Dense Layer	128
Loss Function	cross entropy
Optimizer	Adam

Model Architecture of Multiclass Classification

- Fine-tune pre-trained BERT model from Hugging Face
- Add more text and labels for training
- BertForSequenceClassification
- AdamW optimizer
- Learning rate: 2e-5
- Loss Function: Cross Entropy
- Save best model based on validation loss



Conclusion

Machine Learning

Best accuracy score for each model in binary classification task

Logistic Regression			SGD			SVM		
Hash	TF-IDF	FT	Hash	TF-IDF	FT	Hash	TF-IDF	FT
0.5830	0.6197	0.5337	0.5843	0.6182	0.5283	0.5768	**0.6200**	0.5351

Decision Tree			Random Forest			LightGBM		
Hash	TF-IDF	FT	Hash	TF-IDF	FT	Hash	TF-IDF	FT
0.5644	0.5841	0.5304	0.5832	0.6112	0.5514	0.5901	0.6027	0.5454

BernoulliNB			MLP		
Hash	TF-IDF	FT	Hash	TF-IDF	FT
	0.6007		0.5634	0.6106	0.5336

Results of the 1st sub task of binary classification
(First sub task: test on Kirk's test dataset)

	Hashing		TF-IDF		FastText	
	Acc (↑)	F1 (↑)	Acc (↑)	F1 (↑)	Acc (↑)	F1 (↑)
Logistic Regression	0.5734	0.5729	0.5919	0.5916	0.5413	0.5276
SGD	0.5565	0.5536	0.5902	0.5901	0.5194	0.3418
SVM	0.5430	0.5430	0.5885	0.5885	0.5379	0.5103
Decision Tree	0.5582	0.5403	0.5750	0.5629	0.5126	0.4896
Random Forest	0.5194	0.5193	0.5919	0.5915	0.5548	0.5476
LightGBM	0.5514	0.5514	**0.5919**	**0.5919**	0.5379	0.5376
BernoulliNB			0.5868	0.5846		
MLP	0.5194	0.5174	0.5885	0.5884	0.5261	0.4947
FastText Model						0.5533

Results of the 2nd sub task of binary classification
(Second sub task: test on the dataset scraped from social media)

	Hash		TF-IDF		FastText	
	Acc (↑)	F1 (↑)	Acc (↑)	F1 (↑)	Acc (↑)	F1 (↑)
Logistic Regression	0.5750	0.5671	0.5900	0.5867	0.5000	0.3333
SGD	0.5850	0.5748	0.5750	0.5711	0.5050	0.3443
SVM	0.5900	0.5850	0.5600	0.5524	0.5000	0.3333
Decision Tree	0.4950	0.3703	0.4800	0.3686	0.5000	0.3333
Random Forest	0.5800	0.5758	0.5550	0.5409	0.5850	0.5585
LightGBM	0.5900	0.5867	0.5250	0.4920	0.5100	0.5098
BernoulliNB			0.5350	0.5146		
MLP	**0.6000**	0.5967	0.5850	0.5686	0.5000	0.3333
FastText Model						0.5950

Results of the 1st sub task of multiclass classification
(First sub task: trained on Kirks train and validation sub-datasets, tested on Kirks test sub-datasets)

	Hash		TF-IDF		FastText	
	Acc (↑)	F1 (↑)	Acc (↑)	F1 (↑)	Acc (↑)	F1 (↑)
Logistic Regression	0.2715	0.2812	0.4165	0.4032	0.1012	0.0244
SVM	0.4030	0.4082	0.4890	0.4542	0.2766	0.2967
Decision Tree	0.3406	0.3445	0.3575	0.3611	0.3693	0.3692
Random Forest	0.3609	0.3357	0.3879	0.3736	0.4435	0.3800
LightGBM	0.4047	0.3887	0.3997	0.3930	0.4165	0.3800
MultinomialNB			**0.5059**	0.4430		
MLP	0.4722	0.4430	0.4992	0.4840	0.4840	0.3690
FastText Model						0.4636

Results of the 2nd sub task of multiclass classification
(Second sub task: trained and tested on our own dataset scraped from social media)

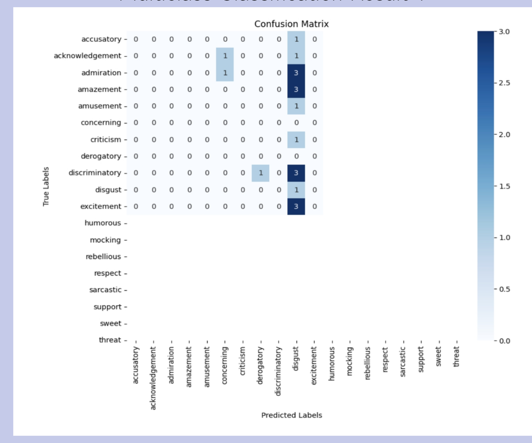
	Hash		TF-IDF		FastText	
	Acc (↑)	F1 (↑)	Acc (↑)	F1 (↑)	Acc (↑)	F1 (↑)
Logistic Regression	**0.2500**	0.1996	0.2250	0.2427	0.1250	0.1417
SVM	0.2000	0.1304	0.1500	0.1330	0.2000	0.2171
Decision Tree	0.2000	0.2121	0.0750	0.0522	0.0500	0.0600
Random Forest	0.1500	0.1378	0.1500	0.1336	0.1250	0.1018
LightGBM	0.1250	0.0902	0.0250	0.0012	0.0750	0.0486
MultinomialNB			**0.2500**	0.2136		
MLP	0.1500	0.0650	0.1250	0.0924	0.100	0.0440

Deep Learning

Deep Learning Binary Classification Result

Model	Embedding Type	Testing Set F1-Score
Bi-LSTM (Phase 1 Table 2.3)	Bi-LSTM Embedding Layer	0.5344
	Pretrained FastText Embedding	0.5545
Bi-LSTM (Phase 2 Table 2.4)	Bi-LSTM Embedding Layer	0.5700
	Pretrained FastText Embedding	**0.5779**
CNN (Phase 1 Table 2.3)	Pretrained FastText Embedding	0.5800
CNN (Phase 2 Table 2.4)		**0.5931**

Multiclass Classification Result I



Different Hyperparameter Testing Values F1-Score Result

Model	(Layer,Embed,Hidden)	3 Times Training F1-Score	Average F1-Score
CNN	(1,32,32)	1st=0.5901 **2nd=0.6222** 3rd=0.5993	0.6038
	(1,32,64)	1st=0.5985 2nd=0.6122 **3rd=0.6249**	0.6118
	(1,50,32)	**1st=0.6228** 2nd=0.6068 3rd=0.6090	**0.6128**
	(1,50,64)	**1st=0.6290** 2nd=0.5989 3rd=0.5945	0.6074
	(2,50,32)	1st=0.5893 2nd=0.5980 **3rd=0.6142**	0.6005
Bi-LSTM	(1,50,32)	1st=0.6005 **2nd=0.6189** 3rd=0.6161	0.6118
	(1,50,64)	**1st=0.6238** 2nd=0.6198 3rd=0.6121	0.6185
	(2,32,32)	1st=0.6145 **2nd=0.6238** 3rd=0.6142	0.6175
	(2,32,64)	1st=0.6317 2nd=0.6222 **3rd=0.6329**	**0.6289**