Team members:
-Mohamed Abdelhamid (m.abdelhamid@innopolis.university)
-Mariam Abuelfotouh (m.abuelfotouh@innopolis.university)
-Ayhem Bouabid (a.bouabid@innopolis.university)

The code of the project can be found here

Introduction:

In this report, we present the results of our Natural Language Processing project undertaken at Innopolis University. Our goal was to build a system that can accurately classify toxic comments in online conversations into six categories: toxic, severe_toxic, obscene, threat, insult, and identity hate.

Overview:

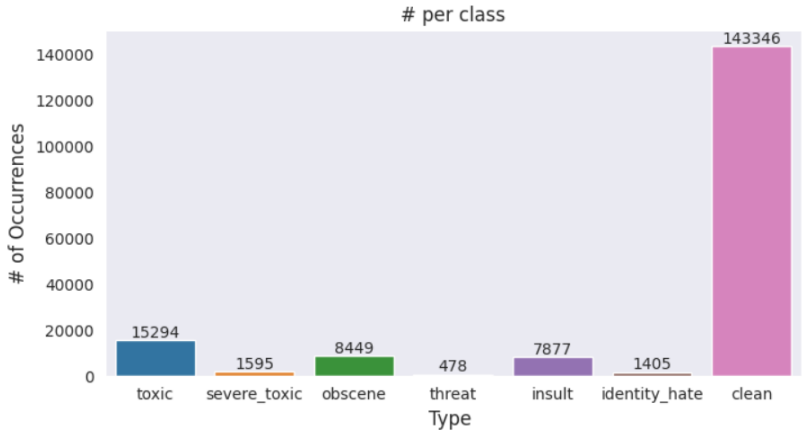
- Data collection and processing
- Architecture and implementation
- Review of methods and models
- Contribution of each member
- Conclusion

Data collection and processing

The data for our project was sourced from a Kaggle competition. The dataset consists of raw text comments posted on social media and includes six labels as outputs. These labels are represented as binary values (1,0), which indicate the presence or absence of the label for a given comment.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
159561	ffd2e85b07b3c7e4	"\nNo he did not, read it again (I would have ...	0	0	0	0	0	0
159562	ffd72e9766c09c97	"\n Auto guides and the motoring press are not...	0	0	0	0	0	0
159563	ffe029a7c79dc7fe	"\nplease identify what part of BLP applies be...	0	0	0	0	0	0

To understand the distribution of the data, we looked at the counts of comments for each of the toxic labels.



We observed that a greater number of comments were labeled as ‘toxic’, ‘obscene’, and ‘insult’, while other labels had relatively few examples. Additionally, despite having a training sample of 111,699 comments, the number of comments labeled across all categories was still low in comparison to the total number of comments. Upon further inspection of the data, we discovered that almost 90% of comments had no toxic labels, indicating that they were clean. This observation highlights the issue of imbalance classification, which is a challenge that needs to be considered during the development of our model.

Another important feature that we noticed while looking at the correlation between the class labels is that there is a good relation between (“toxic”, “insult”), (“toxic”, “obscene”), and (“obscene”, “insult”).

Architecture and implementation

In pursuing our goal of building a good toxic classifier system, we have implemented different models and methods. In this section we will present a brief summary of those models, and their architecture.

Baseline model:

As a baseline model, we decided to use a combination of Naive Bayes and logistic regression. First we tokenized the text, then we used the TF-IDF function from sklearn library for word embedding.

```
def pr(y_i, y):
    p = x[y==y_i].sum(0)
    return (p+1) / ((y==y_i).sum()+1)

x = trn_term_doc
test_x = test_term_doc
```

```
def get_mdl(y):
    y = y.values
    r = np.log(pr(1,y) / pr(0,y))
    m = LogisticRegression(C=4, dual=False)
    x_nb = x.multiply(r)
    return m.fit(x_nb, y), r
```

The **pr** function calculates the probability of a comment being in a particular class, given a set of features. Specifically, it calculates the sum of the feature vectors of all the comments that belong to the class and returns the resulting probabilities.

The **get_mdl** function calculates the weights for the Naive Bayes classifier based on the training data for each label. It first calls the pr function to calculate the probabilities for each class and uses those probabilities to fit a logistic regression model. The resulting model and weights are returned.

Classification using CNNs:

For embedding, we used fastText crawl-300d-2M.vec word vectors.

This is a convolutional neural network (CNN) architecture for text classification. It takes in an input text sequence and applies multiple convolutional filters with different sizes to capture n-gram features. The max-pooling operation is performed on the feature maps obtained from each filter, and the resulting feature maps are concatenated and flattened. Finally, a dense layer with sigmoid activation is used to output probabilities for each class. The model is trained with binary cross-entropy loss and the Adam optimizer.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 200)]	0	[]
embedding (Embedding)	(None, 200, 300)	30000000	['input_1[0][0]']
spatial_dropout1d (SpatialDropout1D)	(None, 200, 300)	0	['embedding[0][0]']
reshape (Reshape)	(None, 200, 300, 1)	0	['spatial_dropout1d[0][0]']
conv2d (Conv2D)	(None, 200, 1, 32)	9632	['reshape[0][0]']
conv2d_1 (Conv2D)	(None, 199, 1, 32)	19232	['reshape[0][0]']
conv2d_2 (Conv2D)	(None, 198, 1, 32)	28832	['reshape[0][0]']
conv2d_3 (Conv2D)	(None, 196, 1, 32)	48032	['reshape[0][0]']
max_pooling2d (MaxPooling2D)	(None, 1, 1, 32)	0	['conv2d[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 1, 1, 32)	0	['conv2d_1[0][0]']

...
Total params: 30,106,502
Trainable params: 30,106,502
Non-trainable params: 0

Classification using LSTM:

For embedding, we used [GloVe](), Global vectors for word representation.

This is a neural network model for multi-label text classification. It uses an embedding layer to convert the input text into a numerical representation, followed by a Bidirectional LSTM layer to capture the

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 100)]	0
embedding (Embedding)	(None, 100, 50)	1000000
bidirectional (Bidirectional)	(None, 100, 100)	40400
global_max_pooling1d (GlobalMaxPooling1D)	(None, 100)	0
dense (Dense)	(None, 50)	5050
dropout (Dropout)	(None, 50)	0
dense_1 (Dense)	(None, 6)	306

Total params: 1,045,756
Trainable params: 1,045,756
Non-trainable params: 0

sequential nature of the text. The output of the LSTM layer is then passed through a GlobalMaxPool1D layer to extract the most important features from the sequence. The extracted features are then fed into two fully connected Dense layers, each followed by a Dropout layer to avoid overfitting. Finally, a sigmoid activation function is applied to the output layer to generate probabilities for each label. The model is trained using binary cross-entropy loss and the Adam optimizer.

Fine-tuning BERT:

This classifier model uses the BERT pre-trained model as an encoder. The input to the model is a string of variable length, and it first goes through a preprocessing layer that applies the necessary tokenization and formatting to match the BERT input format. The output of the preprocessing layer is then fed to the BERT encoder, which produces two outputs: the pooled output and the sequence output. The pooled output is a fixed-length vector representation of the input text and is used as the input to a dropout layer and a dense layer with 500 neurons and ReLU activation. Finally, the output is fed to a dense layer with 6 neurons and sigmoid activation to perform multi-label classification.

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
text (InputLayer)            [(None,)]                 0         []

preprocessing (KerasLayer)   {'input_type_ids':        0         ['text[0][0]']
                              (None, 128),
                               'input_word_ids':
                              (None, 128),
                               'input_mask': (Non
                              e, 128)}

BERT_encoder (KerasLayer)    {'default': (None,        22458881  ['preprocessing[0][0]',
                              512),                                 'preprocessing[0][1]',
                               'sequence_output':                   'preprocessing[0][2]']
                               (None, 128, 512),
                               'pooled_output': (
                              None, 512),
                               'encoder_outputs':
                               [(None, 128, 512),
                               (None, 128, 512)]}

dropout (Dropout)            (None, 512)               0         ['BERT_encoder[0][3]']

dense (Dense)                (None, 500)               256500    ['dropout[0][0]']

classifier (Dense)           (None, 6)                 3006      ['dense[0][0]']

==================================================================================================
Total params: 22,718,387
Trainable params: 22,718,386
Non-trainable params: 1
```

Review of methods and models

To evaluate the performance of each model, we used the same evaluation method used in the Kaggle competition which is mean column-wise ROC AUC. It represents the average of the individual AUC scores of each predicted column(a label is represented as a column).

	Embedding	Size	Score
Baseline	TF-IDF	-	0.97723
CNN	FastText	30 M	0.98200
LSTM	GloVe	1 M	0.9763
BERT	contextual embeddings	22 M	**0.98269**

As you can see BERT is the best performing model. However, the scores of the models are very close to each other and all of them have decent scores.

Contribution of each member and what we learned

Mohamed:

In this project, I contributed by implementing both the CNN and BERT models. Through this experience, I gained valuable skills in teamwork, effective communication, and task distribution among team members. Furthermore, I deepened my knowledge of using CNNs for NLP tasks and fine-tuning BERT models.

Mariam:
In this project I implemented LSTM. Throughout my work in this project, I gained valuable insights on the different word embeddings and how recurrent (and special types of recurrent) neural networks work.

Ayhem:
In this project, I had the opportunity to apply my data analysis skills in a totally different context: Natural Language Processing. Additionally, I implemented the baseline model which enabled the team to estimate the problem's difficulty and objectively determine the performance of mode complex models.

Conclusion

In conclusion, our project has demonstrated the importance of filtering toxic content to foster a healthy online environment. Through our exploration of various NLP approaches such as Naive Bayes, CNNs, LSTM, and BERT, we found that BERT outperformed the other models with a score of 0.98269. The success of our BERT model highlights the potential for its deployment on social networks to automatically filter out toxic comments. As online communication continues to grow, our project contributes to the ongoing efforts to ensure safe and respectful interactions within digital spaces.