Introduction To Data Science
PROJECT

Sensitive Content Detection For English Learners

Team :
Tran Bao Chi
Nguyen Truong Truong An
Nguyen Thanh Dat
Hoang Tran Nhat Minh
Do Hoang Long

Lecturer :
Prof.Than Quang Khoat

Data Science 2022

Abstract

With the growth of the Internet and the number of websites, online newspapers, and social media sites, Enlgish learners using Internet have recently been exposed to a large amount of inappropriate online content, such as insult and harassment comments, political and religious issues that are widely promoted, as well as violent and upsetting war news that aren't really appropriate for everyone. To cope with this, our team is working to develop AI models that can recognize and alert users when a term in an article they are reading is rude, inflammatory, or tries to spread information on a subject, and let them decide whether they should read it or not.

1 Introduction

Nowadays, most people, especially children and the young, have easy and free access to a lot of websites for learning resources that are supportive compared to brick-and-mortar classes. However, they are vulnerable to severe insults and exaggerated news. Moreover, they are easily misled by many falsehoods or, more seriously, by harmful and unverified instructions. Although people have the right to free speech and can access most types of articles on the Internet, censoring or denying access to sensitive content is essential on many platforms, such as social media or search engines.

With the scope of this Data Science subject, our report summarily contribute to detecting sensitive content for English learners by:

1. Data Preparation:
   Because of unavailability of dataset for sensitive contents, we experiment annotating with competent English learners annotators and annotating methods for the sake of labeling time.

2. Proposed methods:
   We also set up a pipeline for modelling and evaluating models to detect the potentially harmful contents.

2 Methods

2.1 Background

Given the textual nature of dataset, we rely on 2 classes of language models (LM): Recurrent Neural Network (RNN) and Transformer. Additionally, we Random Forest in classification task with TF-IDF feature extraction.

2.1.1 RNN

With $x_1, x_2, ..., x_n$ as input sequence, RNN is sequentially formulated as:

$$P(x_t \mid x_{t-1}, \ldots, x_1) \approx P(x_t \mid h_{t-1}) \tag{1}$$

where h_t is hidden state at timestep t

$$h_t = f(x_t, h_{t-1}). \tag{2}$$

The 2 most popular variants of RNN (Fig.3):

1. GRU: Gating models with 2 extra gates (*Update* and *Reset*) for controlling the long-term and short-term memory, and formulated as:

$$\mathbf{R}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xr} + \mathbf{H}_{t-1} \mathbf{W}_{hr} + \mathbf{b}_r), \tag{3}$$

$$\mathbf{Z}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xz} + \mathbf{H}_{t-1} \mathbf{W}_{hz} + \mathbf{b}_z), \tag{4}$$

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{hh} + \mathbf{b}_h), \tag{5}$$

Model $\mathrm{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t$.

LSTM: Gating models with 3 extra gates (*Forget*, *Input* and *Output*) for controlling the long-term and short-term memory, and formulated as:

$$\mathbf{I}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \mathbf{b}_i), \tag{6}$$

$$\mathbf{F}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \mathbf{b}_f), \tag{7}$$

$$\mathbf{O}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \mathbf{b}_o), \tag{8}$$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xc} + \mathbf{H}_{t-1} \mathbf{W}_{hc} + \mathbf{b}_c), \tag{9}$$

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t. \tag{10}$$

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t). \tag{11}$$

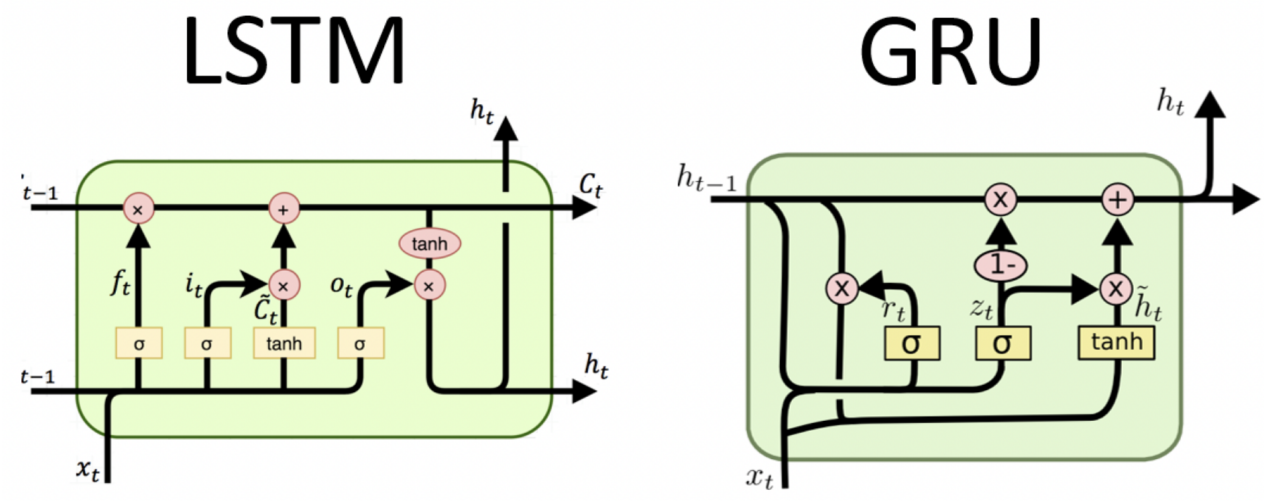


Figure 1: RNN models

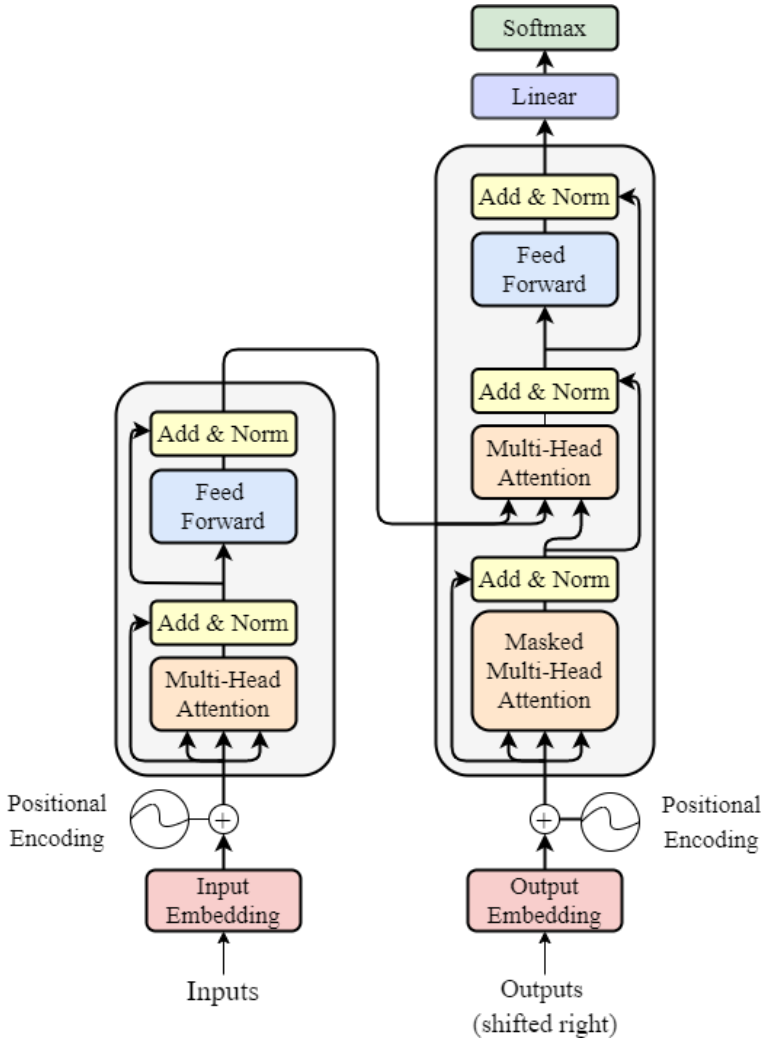


Figure 2: Transformer architecture

2.1.2 Transformer Language Models

With $x_1, x_2, ..., x_n$ as input sequence, Transformer utilize the self-attention mechanism:

$$\mathbf{y}_i = f(\mathbf{x}_i, (\mathbf{x}_1, \mathbf{x}_1), \ldots, (\mathbf{x}_n, \mathbf{x}_n)) \in \mathbb{R}^d \tag{12}$$

by stacking Encoder and Decoder layers:

Inspired by Transformer design, many Pretrained Language Models was designed and trained on large datasets:

1. BERT (Bidirectional Transformers):
   Transformer-Encoder pretrained on 2 tasks: Next Sentence Prediction (NSP), Masked Language model (MLM) capables of capturing contextual representation of text sequences (Fig.3)

2. RoBERTa:
   Robustly pretrained BERT with additional mechanisms: Dynamic Masking, Larger Batch

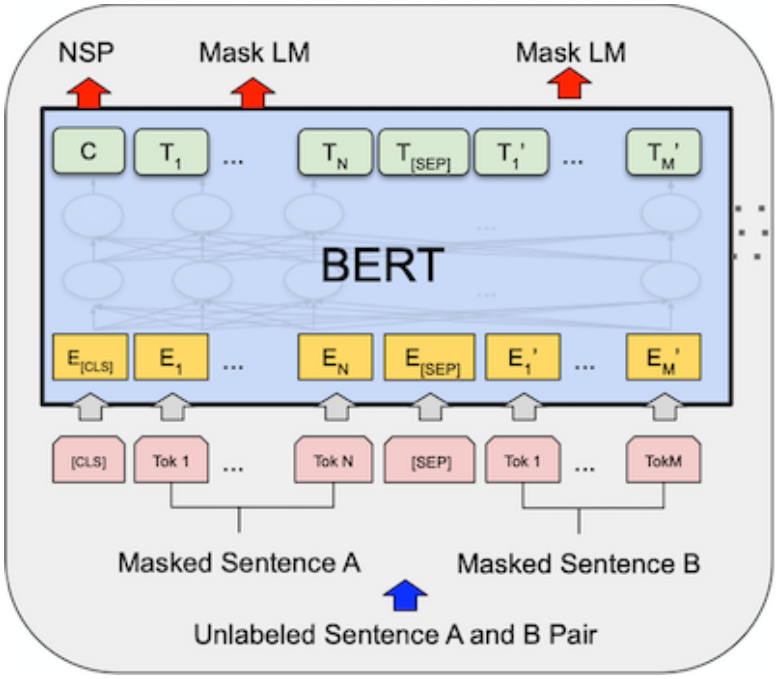


Figure 3: BERT

size, Parameter space, using pair of sentences in same document while get rid of Next Sentence Prediction objective function.

2.1.3 TF-IDF (Term Frequency-Inverse Document Frequency)

In a large text corpus, some words will be very present (e.g. "the", "a", "is" in English) hence carrying very little meaningful information about the actual contents of the document. If we were to feed the direct count data directly to a classifier those very frequent terms would shadow the frequencies of rarer yet more interesting terms.
In order to re-weight the count features into floating point values suitable for usage by a classifier it is very common to use the **tf-idf** transform. tf means term-frequency while $tfidf$ means term-frequency times inverse document-frequency:

$$\mathbf{tfidf(t,d) = tf(t,d) \times idf(t)} \tag{13}$$

$tf(t,d)$ is the number of times the word t occurs in a document d.
$idf(t)$ is employed to determine a word's importance. When calculating the frequency of occurrence tf, the words are considered equally important. Although they are often used, some words are not necessary to convey the document's content. Therefore, we must use idf to lessen the importance of such words.

$$\mathbf{idf(t)} = \log \frac{\mathbf{n}}{\mathbf{df(t)}} + \mathbf{1} \tag{14}$$

where n is the total number of documents in the document set, and $df(t)$ is the number of documents in the document set that contain term t.
The resulting **tf-idf** vectors are then normalized by the Euclidean norm

$$v_{norm} = \frac{v}{||v||^2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + ... + v_n^2}} \tag{15}$$

2.1.4 Random Forest

Random forests are an ensemble method used for classification. In Random Forest, we grow multiple trees as opposed to a single tree in Decision Tree model. One of the major concerns with decision trees is overfitting, which results in an extremely poor predictive model. Adding more trees to the random forest eliminates overfitting and results in a much better predictive model. Each tree offers a classification to categorize a new object based on characteristics, and we say the tree "votes" for that class. The categorization with the highest votes is selected by the forest (over all the trees in the forest). Random forest algorithms have three main hyper-parameters, which need to be set before training. These include node size, the number of trees, and the number of features sampled.

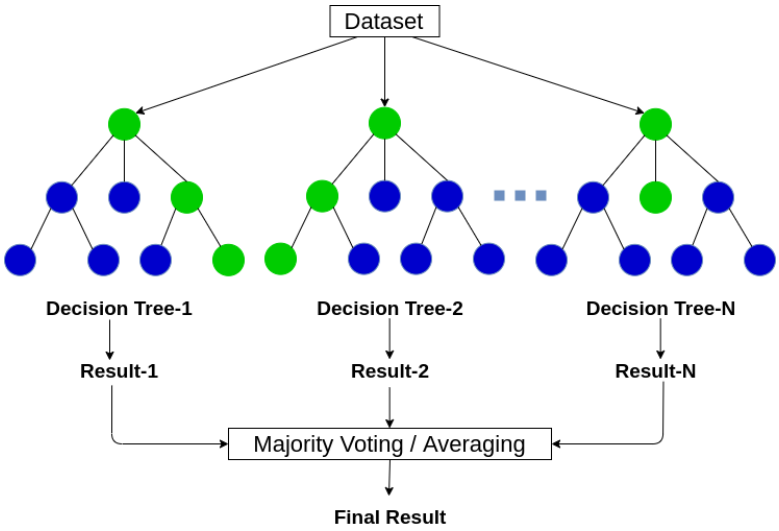


Figure 4: Random Forest

We will employ Random Forest in this problem after TF-IDF model training. Following the Tf and idf-based processing approach, the input text segments are transformed into vectors of the same dimension, where each word represents a dataset feature.To evaluate the quality of a split, we will use the Gini function.

$$\mathbf{G} = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) \tag{16}$$

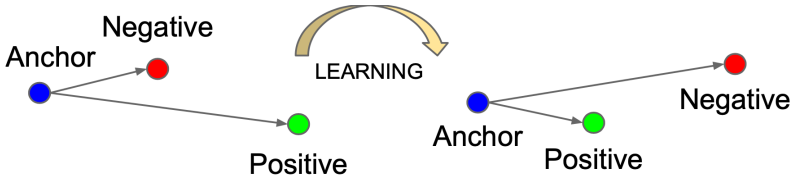
p_{mk} represents the proportion of observations in the m_{th} region from the k_{th} class. In essence, the Gini function is a measure of variance. The higher the variance, the more mis-classification there is. Therefore lower values of the Gini criterion yield better classification.

2.1.5 Constrastive Learning

The goal of contrastive representation learning is to learn such an embedding space in which similar sample pairs stay close to each other while dissimilar ones are far apart. Contrastive learning can be applied to both supervised and unsupervised settings.
Here we apply N-tuple loss with in-batch negatives

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+/\tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i/\tau)}$$

(a) **N-tuple loss**



(b) **CL idea**

Figure 5: Contrastive Learning

2.2 Data Preparation

2.2.1 Active Learning

Active learning is a form of semi-supervised machine learning where the algorithm can choose which data it wants to learn from. With this approach, the program can actively query an authority source, either the programmer or a labeled dataset, to learn the correct prediction for a given problem. Here we apply 3 methods measuring uncertainty for comparison for n unlabeled pool with C number of classes:

1. Random sampling
   Select sample randomly in unlabeled pool

$$i_{RS} \sim U(0, n-1) \tag{17}$$

2. Entropy sampling
   Select the samples with top highest entropy of prediction:

$$i_{ES} = argmax_k \sum p_i log(p_i) \tag{18}$$

3. Least Confidence

$$i_{LC} = argmax_k(1 - max(p_i)) \tag{19}$$

$$s.t \quad i_{RS} \in \{0, 1, ..., n-1\}, i \in \{0, 1, ..., C-1\},$$

2.2.2 Augmentation

Data augmentation is a technique of artificially increasing the training set by creating modified copies of a dataset using existing data. It includes making minor changes to the dataset or using deep learning to generate new data points. We leverage training dataset by 2 sophisticated methods with uniform probability.

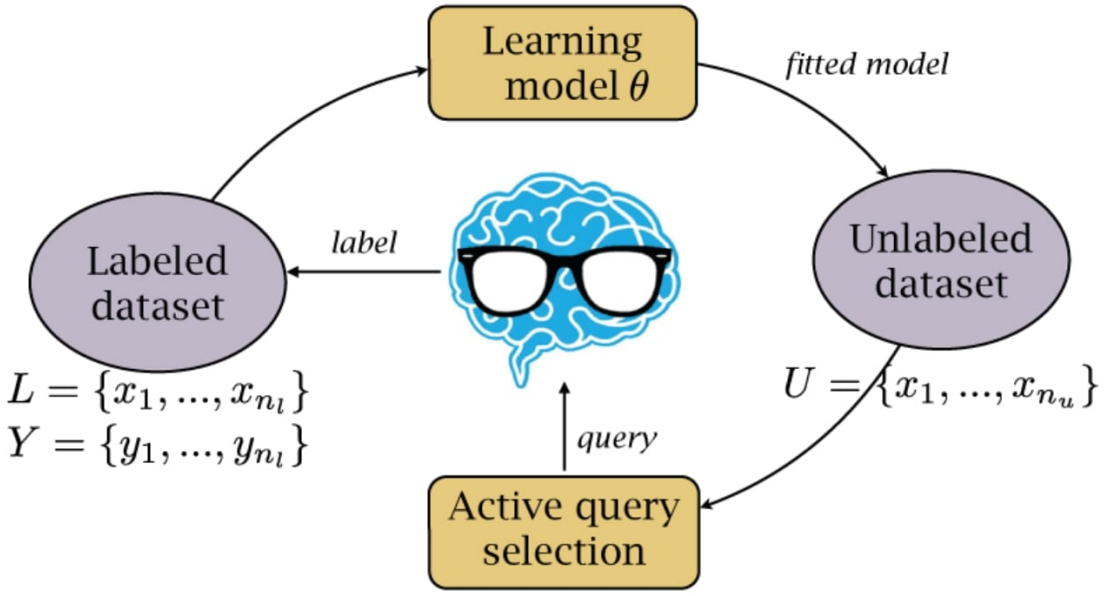


Figure 6: Active Learning

1. Back translation:
   We add the back-translated version from German to English.

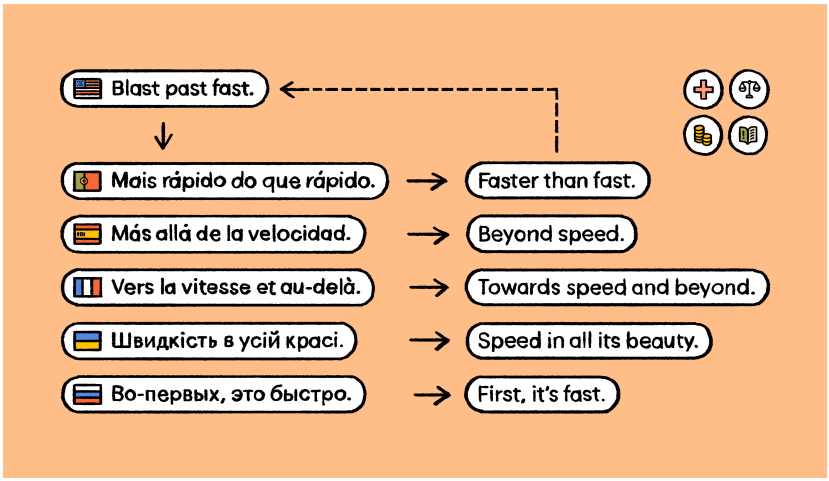
2. Contextual Word Embedding
   We add the top-1 nearest neighbor in embedding pool by utilizing BERT model.

2.3 Text classification

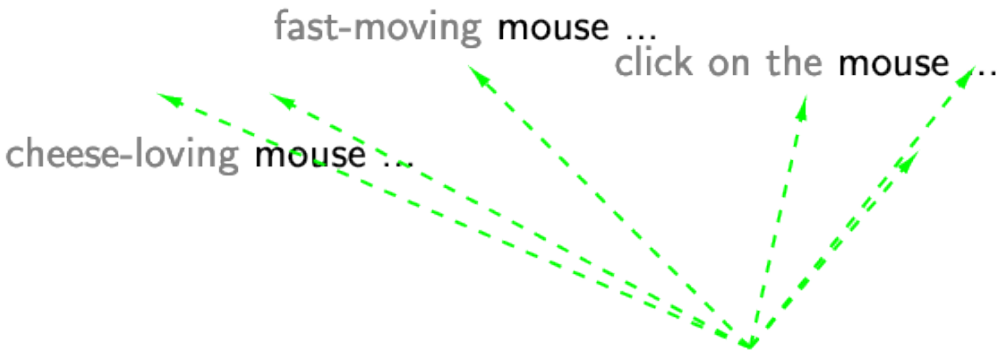
We comply with common steps when applying Machine Learning models, for leveraging deep learning language models, the problem is formulated as:

$$z_j = LM(x_j); \quad p(y_i|x_j) = \frac{e^{W_i z_j + b_i}}{\sum e^{W_k z_j + b_k}} \tag{20}$$

with label y_i, input sentence x_j, and z is hidden representation which is hidden state h in RNN models and $[CLS]$ token in Transformers models



(a) **Back translation**



(b) **Contextual embedding**

Figure 7: Augmentation methods

3 Experiments

In this report we training each module on designated datasets for IDSF and QA, and evaluated in common metrics for each task afterwards.

3.1 Main pipeline

Our classification pipeline is broken down into following steps:

1. Crawling
2. Annotating
3. Preprocessing
4. Feature extraction or Tokenization
5. Modelling
6. Evaluation

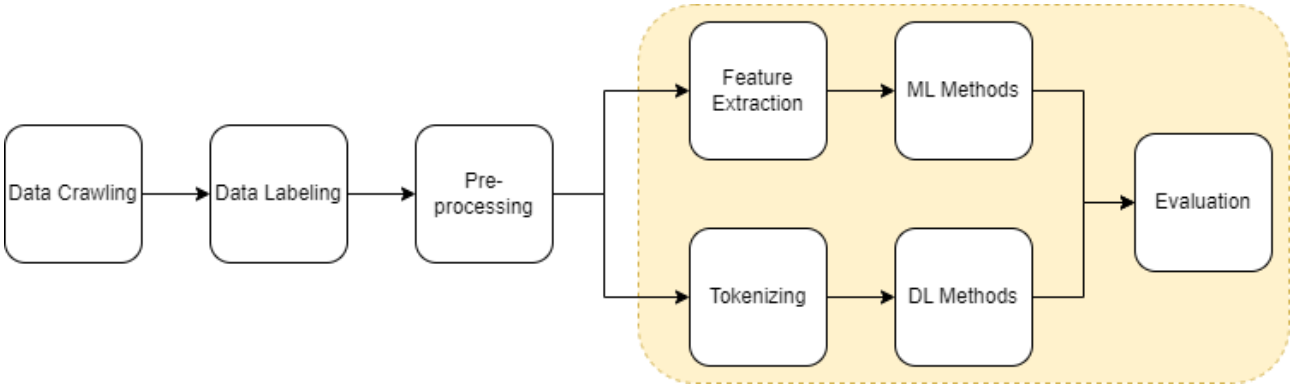


Figure 8: Main Pipeline

3.2 Data

3.2.1 Annotating

By apply Active Learning in Annotating, by starting with 1000 labeled for testing, we can witness the accuracy boost relative to the number of samples annotated. Thouugh perform worse at nascent stages of both (Due to small number of samples), We witness the outperformance of 2 methods Entropy Sampling and Least Confidence over Random Sampling. Because of time strain and experiment scope, we cease annotating process at around 1500 training samples.

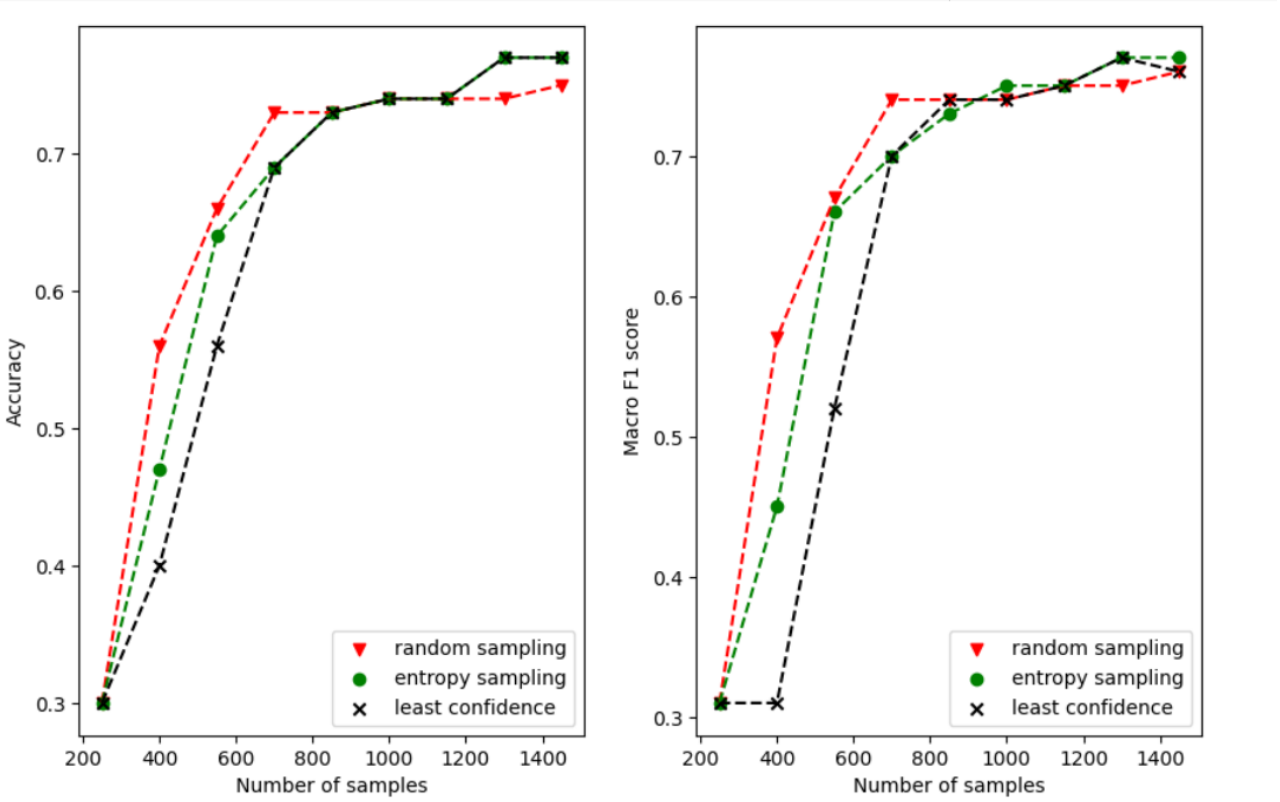
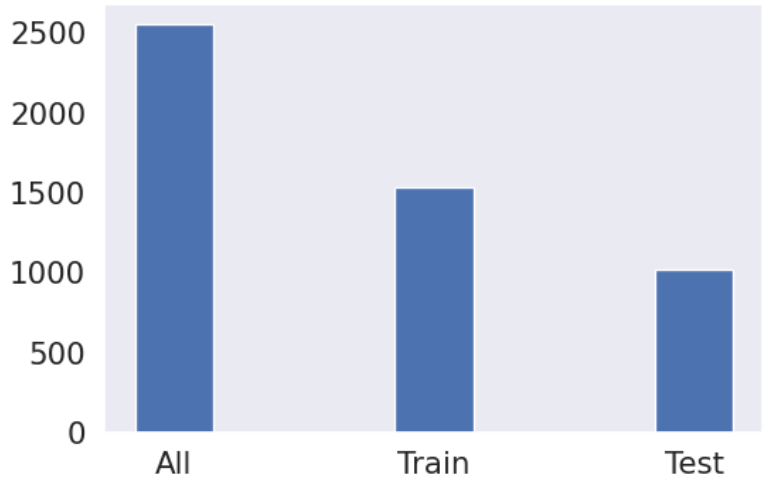


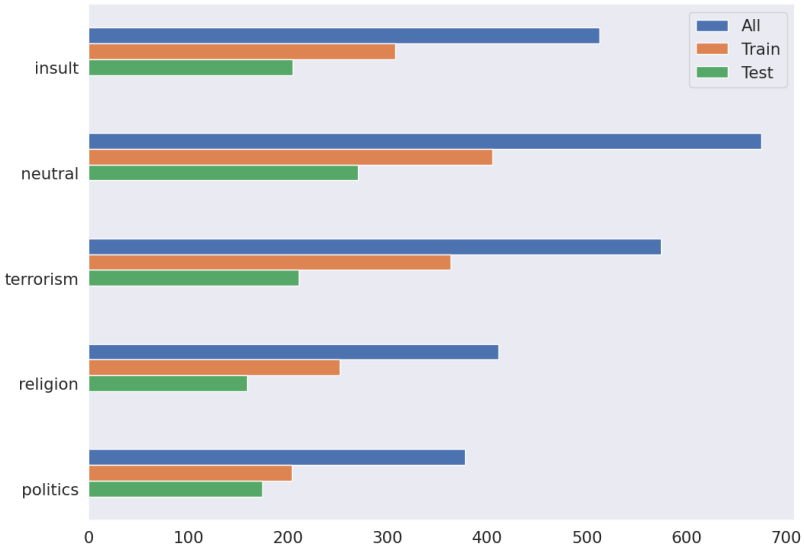
Figure 9: Active Learning

3.2.2 Dataset

By active learning process, we end up with dataset split with around 1500 training samples and 1000 testing samples. Additionally, we augment our training dataset with visualized statistics.



(a) **Number of samples**

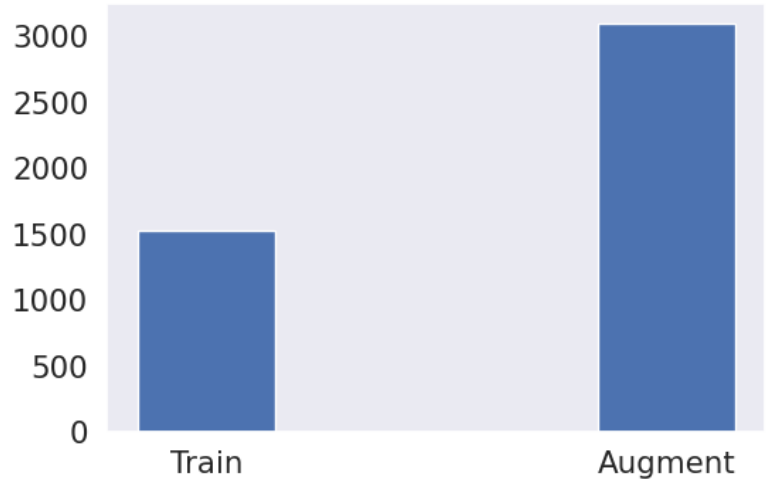


(b) **Class distribution**

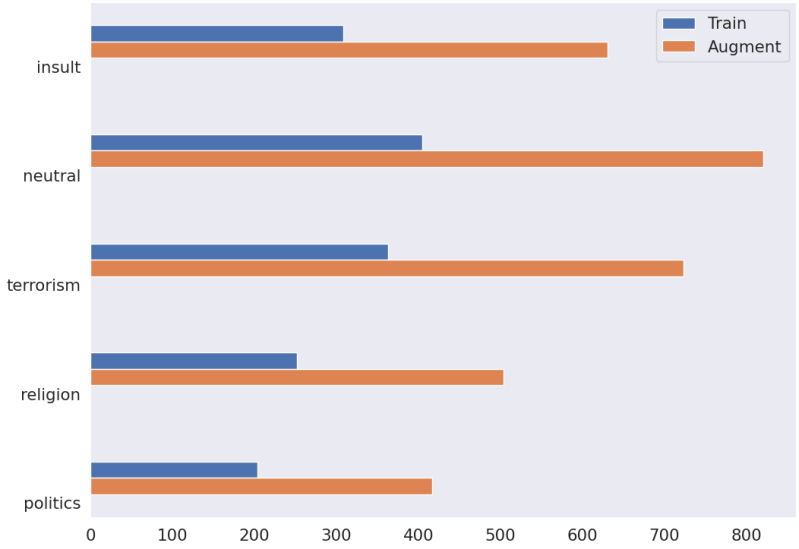
Figure 10: Data split

3.3 Results

We trained RNNs and Transformers modelby and tuning defined hyperparameters space with AdamW optimizer and warm-up scheduler. We can observe the outperformence of pretrained LLM over Random forest and RNN methods, while LSTM even perform worser then Random Forest method. Roberta model with better pretrain strategy reached slightly beter result then pretrained Bert model.



(a) **Number of samples**



(b) **Class distribution**

Figure 11: Augmentation statistics

Model	Acc. (%)	F1 Score
Random Forest	71.2	0.71
LSTM	70.0	0.69
Bert	76.5	0.77
XLM-RoBERTa	76.7	0.77
Fnet	55.8	0.49

Table 1: Classification results

3.4 Ablation Study

We examine the advantages of Contrastive Learning and Data Augmentation. By double the number of samples in training data or applying contrastive learning, we observe the increase

in accuracy by small margin. Here we use BERT model as testbed. Leveraging Augmentation method and robust learning strategy Contrastive Learning truly expand testbed capacity, we will dive deeper into this techniques further in future works.

Method	✓	×
Augmentation	77.3	76.5
Contrastive Learning	77.0	76.5

Table 2: Ablation Study

4 Conclusions and Future work

In this report, we has:

1. Data Prepration:
   we experiment annotating with competent English learners annotators and annotating methods for the sake of labeling time because of unavailability of dataset for sensitive contents.

2. Proposed methods:
   We also propose a pipeline for modelling and evaluating models to detect the potentially harmful contents. We compared different models and have ablation study for data augmentation and Contrastive Learning

Because of the nascence of our work as well as the fact that the process of improving mentioned problems is long-term process. We leave several points for future works:

1. Annotating more data

2. Trying out sophisticated Active Learning methods

3. Experimenting different advanced models

4. Trying Multi-class approach

We also in deep gratitude towards many researchers and websites [3] [2] [4] [1]

References

[1] Varun et al. *Data Augmentation Using Pre-trained Transformer Models.*

[2] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach.*

[3] Many authors. *Dive into Deep Learning.*

[4] David S.Batista. *Conditional Random Fields for Sequence Prediction.*