

# Toxic Span Detection

CS779 - Statistical NLP

## Group 13

<i>Samyak Jain</i>	<i>180661</i>	<i>samyakj@iitk.ac.in</i>
<i>Archit Bansal</i>	<i>180134</i>	<i>architb@iitk.ac.in</i>
<i>Abhay</i>	<i>180014</i>	<i>kabhay@iitk.ac.in</i>

# Problem Description

As a complete submission for the Shared Task, systems will have to extract a list of toxic spans, or an empty list, per text. As toxic span we define a sequence of words that attribute to the text's toxicity. Consider, for example, the following text:

"This is a stupid example, so thank you for nothing a!@#!@."

Systems are then expected to return the following list for this text:

- [10,11,12,13,14,15,51,52,53,54,55,56]

Official Links: [Codalab](#) and [Google Site](#)



# Evaluation Matrix

The final evaluation is based on average  $F1$  score:


$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)}$$

$$P^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_{A_i}^t|}$$


$$R^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_G^t|}$$



# Motivation

1. Moderation of content is crucial to promote healthy online discussions.
  2. Although several toxicity (abusive language) detection datasets and models have been released, most of them classify whole comments or documents, and do not identify the spans that make a text toxic. But highlighting such toxic spans can assist human moderators (e.g., news portals moderators) who often deal with lengthy comments, and who prefer attribution instead of just a system-generated unexplained toxicity score per post..
  3. The evaluation of systems that could accurately locate toxic spans within a text is thus a crucial step towards successful semi-automated moderation.
- 

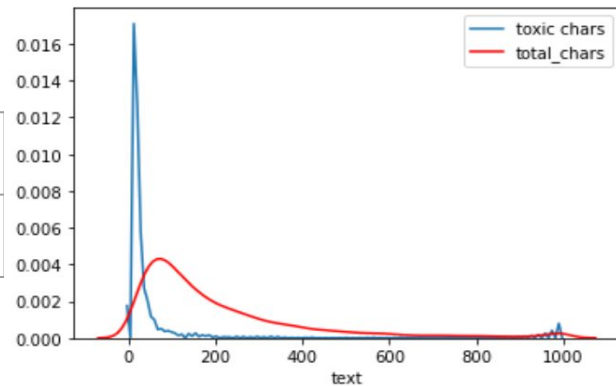
# Related Work

1. Existing datasets and models in the field of offensive content classify whole comments or documents as toxic(or not), and **do not identify the spans** that make a text toxic.
  2. [Pavlopoulos\(2017\)](#) showed that RNN-based moderation methods can be improved by adding user embeddings, user type embeddings, user biases, or user type biases for task of content moderation.
  3. Many recent contributions came from the [OffensEval](#) task. It involved three subtasks Identification, Categorization and Target Identification.
  4. [NULI](#)(2019) team used BERT and got SOTA results for the subtask 1 while [NLPR@SRPOL](#)(2019) used ensemble models (LSTM, Transformer, OpenAI's GPT, Random forest, SVM)) and got decent results on all subtasks.
- 

# Dataset

- Annotated civil\_comments dataset

Toxic Spans	Text
[0,1,2,3]	<b>Damn</b> , a whole family. Sad indeed.



	words	chars
avg	35.9	204.5
max	192	1000
min	1	4

Dataset/Split	Total
train	6531
dev	794

# Tokenization

Used TreeBankWordTokenizer : Rule Based

- Treating punctuation chars as individual tokens.
- Split commas and quotes when followed by whitespaces.
- Retained span of each token : (*“he is playing”*) => [*“he”*, *“is”*, *“playing”*], [(0,2), (3,5), (6,13)]
- Split standard contractions: eg *“don’t”* to [*“do”*, *“n’t”*]
- Words like *“jacka\*\*”*, *“idi\*t”* are retained.



# Preprocessing

**All of the below preprocessing were applied after tokenization.**

- Removed numbers, special chars, lowercasing
- Removed emojis (not marked toxic by annotators)
- Short words expanding ( **don't** => [do, n't] =>[do, not] )
- Removed punctuations
- Labelled toxic tokens as 1 (remaining labelled as 0).

**Improvements (Future work) :**

- Using punctuations as features.
- Spell correction
- Handling intentionally misspelled words eg . **idi\*t, f\*\*k**



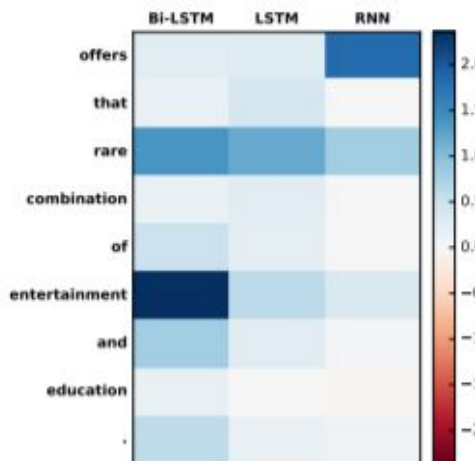


# Baseline-1 (Representation Erasure & LIME)

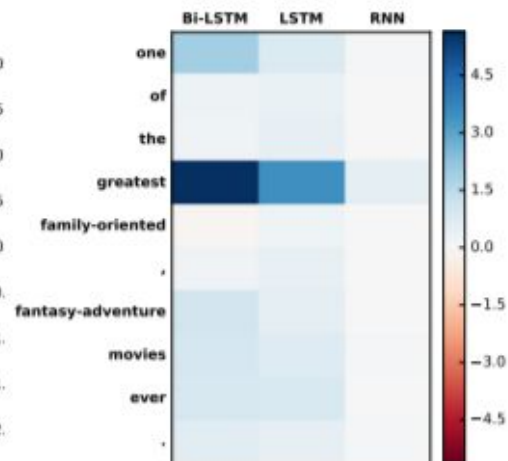
- Understanding the importance of each word towards the classification of a text as carrying a particular sentiment
- Representation Erasure - Looking at the change in log likelihood of the label of the text upon removal of a specific word

$$I(d) = \frac{S(e, c) - S(e, c, \neg d)}{S(e, c)}$$

- LIME - Pre-built Python library to understand neural networks through a model agnostic approach



(c) Strong positive

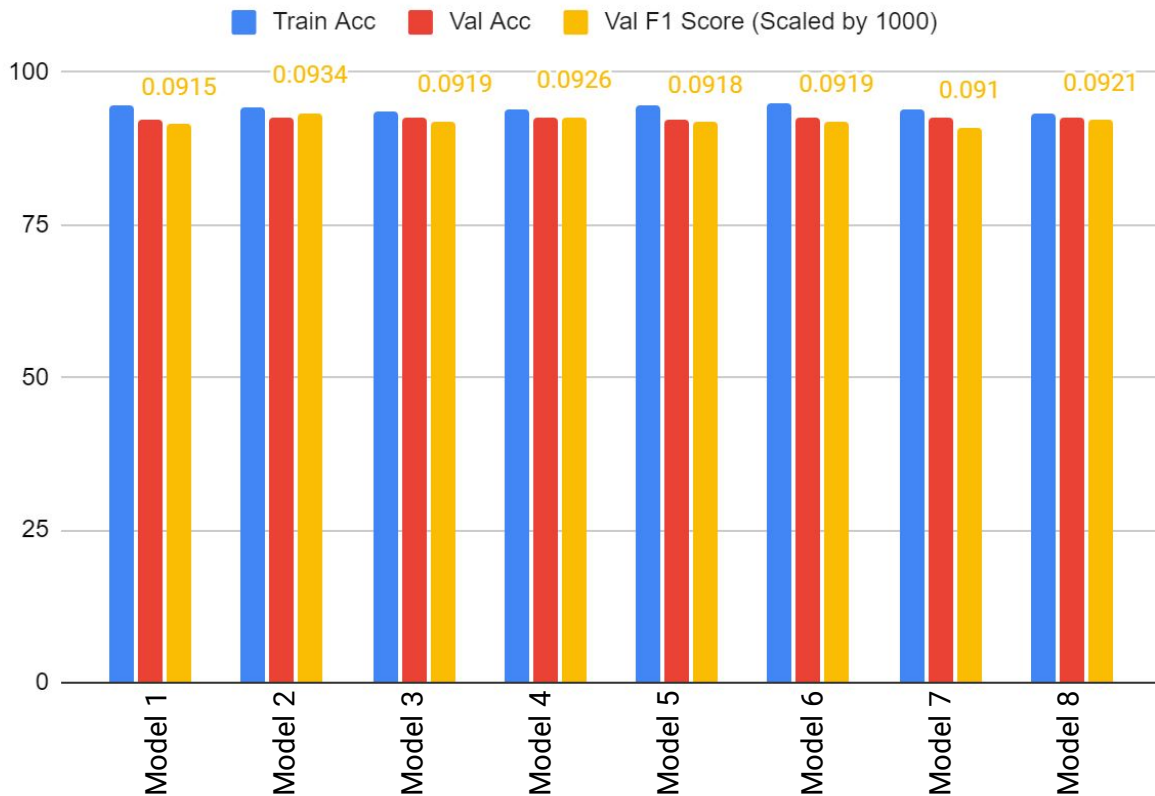


(d) Strong positive

Representation Erasure : <https://arxiv.org/pdf/1612.08220.pdf>

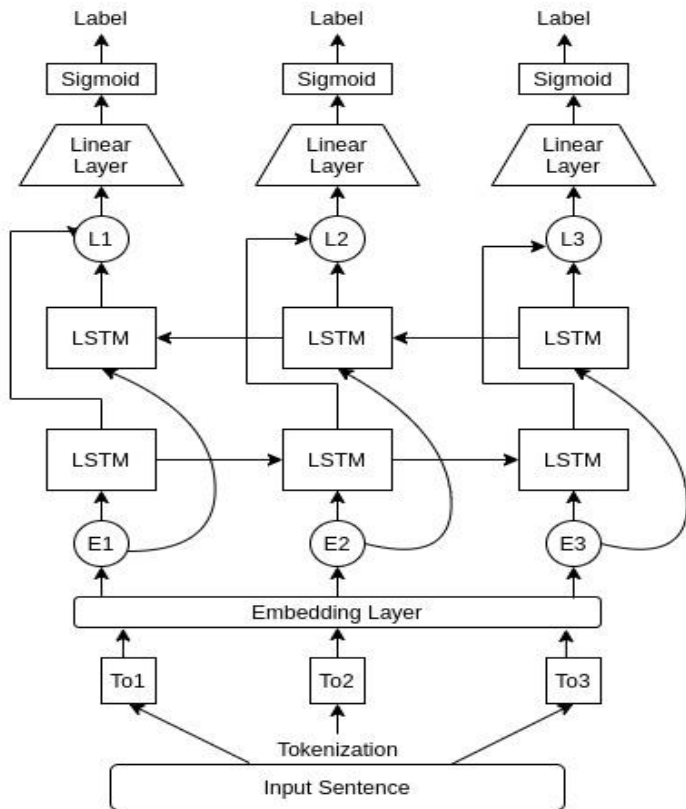
LIME : <https://arxiv.org/pdf/1602.04938.pdf>

# Baseline-1 (Representation Erasure & LIME)



No. : BiLSTM Layer	Dropout	Glove Embeddings	Other Changes
1	0.5	6B.100d	
2	0.5	6B.100d	
1	0.5	twitter.27B.200d	
1	0.7	twitter.27B.200d	
1	0.7	twitter.27B.200d	Unk token not initialized to 0
1	0.7	twitter.27B.200d	Lower case dataset
2	0.5	twitter.27B.200d	Lower case dataset
2	0.5	6B.100d	Lower case dataset

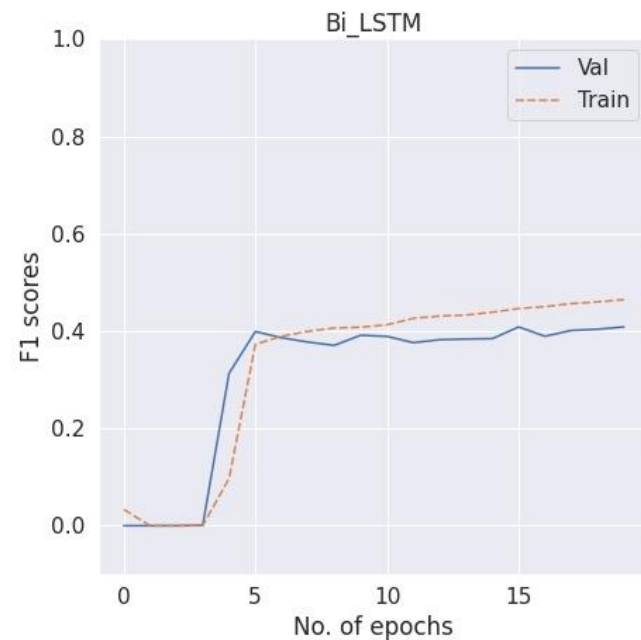
# Baseline-2 : BiLSTM



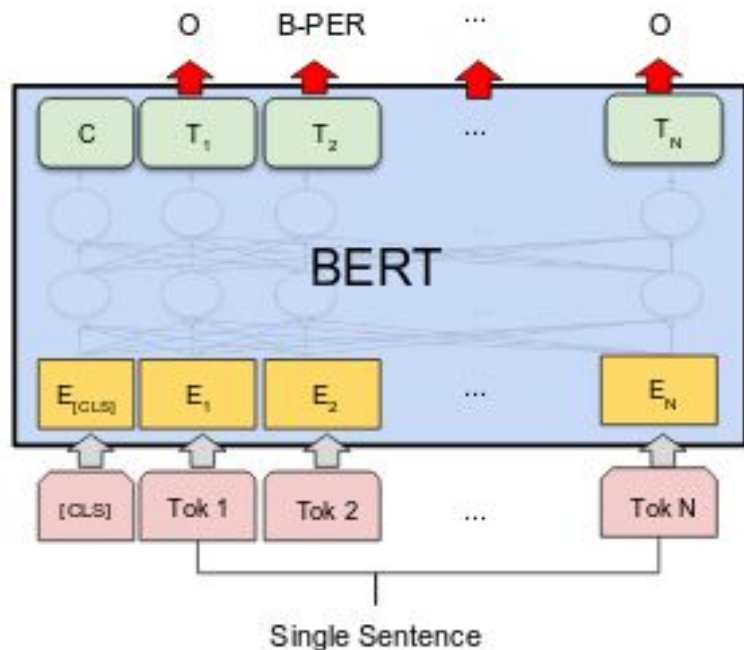
1. Bi-LSTM has been used in tons of natural language processing task, such as sentiment classification, neural translation, language generation etc.
2. We tried Bi-LSTM with various settings, results summarised in the next slide.
3. Calculated the F1 score comparing the labels of tokens. (We want to improve these scores first.)

# Bi-LSTM Results

Epoch	Bi-LSTM Layers	Hidden Dim	Pretrained Embeddings	Train F1 score	Val F1 score
10	2	64	NA	0.3	0.25
10	2	64	glove.6B.300d	0.365	0.32
10	3	64	glove.twitter.27B.200d	0.412	0.38
10	2	64	glove.twitter.27B.200d	0.435	0.4137
20	2	64	glove.twitter.27B.200d	0.464	0.4082
10	2	128	glove.twitter.27B.200d	0.423	0.401



# Baseline-3 BERT



Source :

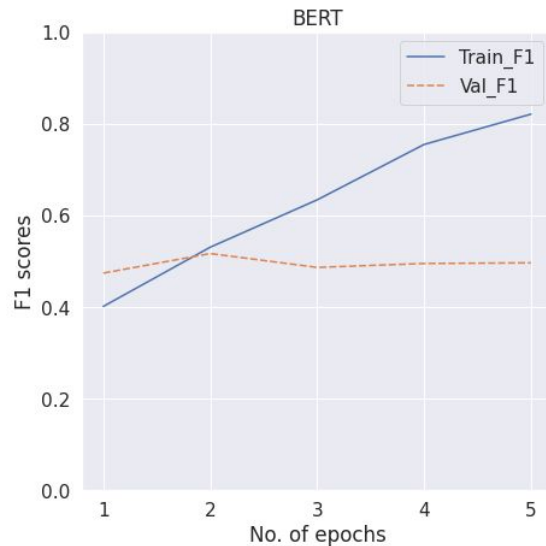
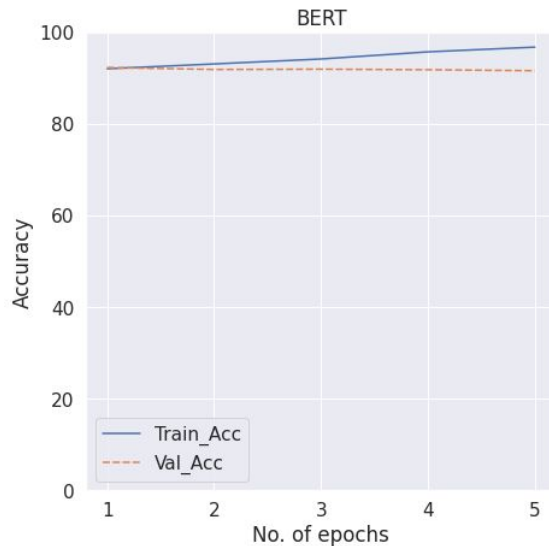
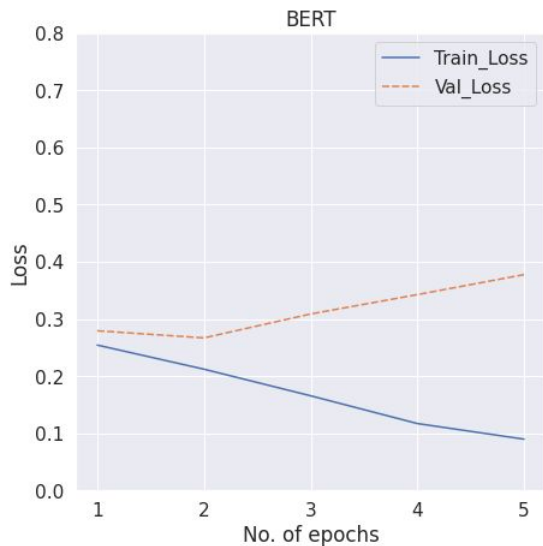
<https://www.depends-on-the-definition.com/named-entity-recognition-with-bert/>

- Motivation from Bert for Name Entity Recognition Task
- Bert Model with a token classification head on top (a linear layer on top of the hidden-states output)
- For the initial baseline, we have used the Bert-Base-Cased pretrained model

BERT : <https://arxiv.org/pdf/1810.04805.pdf>  
HuggingFace BertForTokenClassification : [Link](#)

# Baseline-3 BERT

- None of the pretrained layers were frozen
- Weight decay was used to tackle regularization
- Learning Rate was also decayed over epochs
- Max Length of Sequence : 100



# Future Directions

1. Improvising BERT
  - a. Add Bi-LSTM head
  - b. Try different preprocessings and pretrained models.
  - c. Use different Tokenizer (BPE)
2. RoBERTa/XLNet (expected to give better results than BERT)
3. Idea of Zero- Shot Learning (Fine tune Bert on Sequence Labelling task).
4. Try with different loss functions (A.T loss).
5. Graph Convolutional Networks for Node Classification (If time permits)



# Timeline

## Current Position:

### Three Baseline Models :

1. Representation Erasure
2. Bi-LSTM
3. BERT

## Contributions:

1. Dataset Cleaning and Analysis (Samyak)
2. BiLSTM and modifications (Abhay)
3. Representation Erasure (Archit)
4. Bert (Archit)

## Future Plans :

1. Improving BERT - Oct '20
2. RoBerta/XLNet - Nov First Week
3. Zero Shot Learning - Nov '20





The background is a solid pink color. In the top right corner, there is a decorative arrangement of geometric shapes: a light pink triangle pointing down-right, a dark pink square, and another light pink triangle pointing up-right, all partially overlapping. A diagonal line of a slightly darker pink shade runs from the top right towards the bottom left, creating a sense of depth and movement.

THANKS