Intro to NLP: Assignment 2 Offensive Language Detection

Content warning: this assignment contains an analysis of offensive language examples.

In this assignment, we will work with the OLIDv1 dataset, which contains 13,240 annotated tweets for offensive language detection. The detailed description of the dataset collection and annotation procedures can be found here. This dataset was used in the SemEval 2019 shared task on offensive language detection (OffensEval 2019).

We will focus on **Subtask A** (identify whether a tweet is offensive or not). We preprocessed the dataset so that label '1' corresponds to offensive messages ('OFF' in the dataset description paper) and '0' to non-offensive messages ('NOT' in the dataset description paper).

The training and test partitions of the OLIDv1 dataset (olid-train.csv and olid-test.csv, respectively) can be found here.

You submit a **pdf** of this document, the format should not be changed.
Your analyses should be conducted using **python 3.8**.
You submit a **zip**-file containing all your code.

Each team member needs to be able to explain the details of the submission. By default, all team members will receive the same grade. If this seems unjust to you, provide an extra statement indicating the workload of each team member.

Total points: 20
Structure:

- Part A: Fine-tune BERT for offensive language detection (7 points)
- Part B: Error analysis with checklist (13 points)
- Bonus tasks: options for obtaining a grade > 8

Fill in your details below:
Group number: 69

Student 1
Name: Don Mani
Student id: 2693434

Student 2
Name: Azhar Shaikh
Student id: 2701842

Student 3
Name: Caroline Hallmann
Student id: 2640914

Part A: Fine-tune BERT for offensive language detection (7 points)

1. **Class distributions (1 point)**

Load the training set (olid-train.csv) and analyze the number of instances for each of the two classification labels.

Class label	Number of instances	Relative label frequency (%)	Example tweet with this label
0	**8840**	**66.77**	**@USER @USER Obama wanted liberals & illegals to move into red states**
1	**4400**	**33.23**	**@USER @USER Go home you're drunk!!! @USER #MAGA #Trump2020 👊🇺🇸👊 URL**

Class label 0 is for non-offensive tweets and 1 for offensive tweets.

2. **Baselines (1 point)**

Calculate two baselines and evaluate their performance on the test set (olid-test.csv):

- The first baseline is a random baseline that randomly assigns one of the 2 classification labels.
- The second baseline is a majority baseline that always assigns the majority class.

Calculate the results on the test set and fill them into the two tables below. Round the results to two decimals.

Random Baseline			
Class	Precision	Recall	F1
0	**0.69**	**0.48**	**0.57**

1	**0.24**	**0.43**	**0.31**
macro-average	**0.47**	**0.46**	**0.44**
weighted average	**0.56**	**0.47**	**0.49**

Majority Baseline			
Class	Precision	Recall	F1
0	**0.72**	**1.00**	**0.84**
1	**0.00**	**0.00**	**0.00**
macro-average	**0.36**	**0.50**	**0.42**
weighted average	**0.52**	**0.72**	**0.60**

3. **Classification by fine-tuning BERT (2.5 points)**

Run your notebook on colab, which has (limited) free access to GPUs.

You need to enable GPUs for the notebook:
- navigate to Edit → Notebook Settings
- select GPU from the Hardware Accelerator drop-down

➢ Install the simpletransformers library: *!pip install simpletransformers*
(you will have to restart your runtime after the installation)
➢ Follow the documentation to load a pre-trained BERT model: ClassificationModel('bert', 'bert-base-cased')
➢ Fine-tune the model on the OLIDv1 training set and make predictions on the OLIDv1 test set (you can use the default hyperparameters). Do not forget to save your model, so that you do not need to fine-tune the model each time you make predictions.
If you cannot fine-tune your own model, contact us to receive a checkpoint.

a. Provide the results in terms of precision, recall and F1-score on the test set and provide a confusion matrix **(2 points)**.

Fine-tuned BERT

Class	Precision	Recall	F1
0	**0.86**	**0.94**	**0.90**
1	**0.78**	**0.61**	**0.69**
macro-average	**0.82**	**0.77**	**0.79**
weighted average	**0.84**	**0.84**	**0.84**

Confusion Matrix: Fine-tuned BERT		
	Predicted Class	
Gold Class	**0**	**1**
0	**580**	**40**
1	**94**	**146**

b. Compare your results to the baselines and to the results described in the paper in 2–4 sentences **(0.5 points)**.

We get 0.79 macro F1 with default parameters using BERT cased trained for one epoch. The best performing model with 0.829 macro F1 in the competition was BERT-uncased trained for 2 epochs. When comparing the results of the BERT model to the random baseline, the BERT model has generated better results in terms of performance. The BERT model outperformed the random baseline model. The results of the BERT model versus the majority baseline shows higher performance of BERT. However, the majority baseline recall of class 0 had the best performance. Overall, BERT had the best performance values and would be in the top competitors of the challenge discussed in the paper.

4. **Inspect the tokenization of the OLIDv1 training set using the BERT's tokenizer (2.5 points)**

The tokenizer works with subwords. If a token is split into multiple subwords, this is indicated with a special symbol.

a. Calculate how many times a token is split into subwords (hint: use model.tokenizer.tokenize()). (0.5 points)

Number of tokens: **387931**

Assumption:

We consider tokens of 2 types: splittable tokens that can be split into subwords and non-splittable tokens that can not be split into subwords. All tokens are subwords. (A token that is not split consists of 1 subword)

Number of tokens that have been split into subwords: **67045**

Example: if ‘URL’ is tokenized by BERT as ‘U’, ‘##RL’, consider it as one token split into two subwords.

b. What is the average number of subwords per token? (0.5 points)

Average number of subwords per token: **1.23**

c. Provide 3 examples of a subword split that is not meaningful from a linguistic perspective. (1 point)

Which split would you expect based on a morphological analysis?

1. Example 1: **#TrudeauMustGo**
2. BERT tokenization: ‘**#', 'T', '##rud', '##eau', '##M', '##ust', '##G', '##o’**
3. Morphologically expected split: **‘#’, ‘Trudeau’, ‘Must’, ‘Go’**

1. Example 2: **groping**
2. BERT tokenization: **'g', '##rop', '##ing**
3. Morphologically expected split: **‘grope’, ‘ing’**

1. Example 3: **warped**
2. BERT tokenization:**’war', '##ped'**
3. Morphologically expected split:**’warp’, ‘ed’**

d. BERT’s tokenizer uses a fixed vocabulary for tokenizing any input (model.tokenizer.vocab). How long (in characters) is the longest subword in the BERT’s vocabulary? (0.5 points)

Length of the longest subword: **18**

Example of a subword with max. length: **Telecommunications**

Part B: Error analysis with checklist (13 points)

Often accuracy or other evaluation metrics on held-out test data do not reflect the actual model behavior. To get more insights into the model performance, we will employ three different diagnostic tests, as described in https://github.com/marcotcr/checklist.

Relevant literature:

- https://homes.cs.washington.edu/~marcotcr/acl20_checklist.pdf
- https://arxiv.org/pdf/2012.15606.pdf

Creating examples from existing datasets via perturbations (10.5 points)

Use a subset of the OLIDv1 test set, which contains 100 instances:
(olid-subset-diagnostic-tests.csv, can be found in the same directory) and run the following tests:

5. **Typos (6 points)** Spelling variations are sometimes used adversarially to obfuscate and avoid detection (Vidgen et al., 2019; subsection 2.2), that is, users introduce typos to avoid their messages being detected by automated offensive language/hate speech detection systems. Let us examine how it influences our offensive language detection model.
   Use checklist to add spelling variations (typos) to the subset (olid-subset-diagnostic-tests.csv) and evaluate the model's performance on the perturbed data. Use a fixed random seed (`np.random.seed(42)`) to facilitate comparison.

   Quantitative analysis:
   - Describe the differences in performance compared to the non-perturbed data (precision, recall, F1-score macro). **(1 point)**

	Precision Macro	**Recall Macro**	**F1-score Macro**
Non-Perturbed	**0.85**	**0.81**	**0.80**
Perturbed	**0.79**	**0.76**	**0.75**

   As seen in the above table, including typos reduced the precision, recall and F1 macro scores.
   - How many messages were identified correctly in the original dataset, but erroneously after the perturbation? **(1 point)**

81 messages were identified correctly in the original dataset. After typo perturbation, 3 messages were incorrectly identified. (i.e. 77 messages identified correctly)

Qualitative analysis:

- Check the add_typos function in checklist pertub.py. How were the typos introduced? **(1 point)**
  Typos are introduced by swapping random characters with its neighbors. Using the default parameters of the function only one random character is swapped with its neighbor. We set typos = 3, which results in 3 random characters being swapped with its neighbors.

- Provide an example of a typo that cannot be produced by this function, but would play a role in offensive language detection. **(0.5 points)**
  Example: FFuck
  We can perturb a typo by repeating a particular character. (In this case 'F')

- Provide 3 examples when the model failed to assign the correct label after perturbation. **(1 point)**

ID	Original text	Perturbed text
92215	#Emmys Well, most of those ex-crackheads are ...	#Emmys W ell, most of those ex-crakcheads are ...
14582	#LifeIsButepu when you have sex anytime you wa...	#LifeIsButepu when you have sex anytimey ou wa...
76135	@USER @USER @USER @USER Beats the hell NJ out	@USER @UESR @USER @USER Beats the hell NJ out

In all the above cases the perturbation resulted in the model misclassifying the messages as non-offensive.

- What is the main source of the erroneous predictions produced by the model (main source of errors caused by typos)? **(1 point)**
  The main source of errors caused by typos is perturbation of swear words.

- How can the model be improved? **(0.5 points)**
  The model can be improved further by having a typo correction module/spell check module before passing the data to the offensive language detector model. The model can also be trained on perturbed data, so that the model learns to classify correctly on perturbed data. This is called Data Augmentation.

6. **Negation (4.5 points)** Offensive language detection models have been shown to struggle with correctly classifying negated phrases such as “I don’t hate trans people” (Rottger et al., 2021; subsection 2.2).
Add negations to the subset and evaluate the model's performance on the perturbed data.
Qualitative analysis:
    - Check the add_negation function in checklist pertub.py. What kind of negations does it produce? **(1 point)**

The function negates verbs by transforming the word into the opposite meaning by the implementation of ‘not’ (`doc[:root_id].text + ' not ' + doc[root_id:].text.`).

The following verbs below were negated as follows:

Original	*Negated form*
Is	**Is not**
Was/were	**Was not/were not**
Do/did	**Don’t/ didn’t**
Can/could	**Can’t/couldn’t**
Am/are	**Am not/are not**
Will/would	**Won’t/wouldn’t**
‘s/’re/’m	**‘S not/’re not/’m not**
Have/had	**Haven’t/hadn’t**

- Look at the created negated sentences, are they linguistically correct? Provide 2–5 examples of linguistically incorrect sentences. **(1 point)**

Example 1: @USER @USER Who the hell doesn't he think he is?

Example 2: JEFF doesn't session : ‘LISTEN TO THE ACLU, ANTIFA, BLACK LIVES MATTER’ IF YOU WANT MORE CHICAGO SHOOTINGS

Example 3: #Capitalism may be not undermined by the displacement of menial laborers by #AI should conservatives therefore inoculate themselves against knee-jerk reactionism against the notion of the living wage"? #economics #marxism #marx #robotics #communism #socialism #socialismkills"

- Check the first 10 negated messages. For which of these negated messages should the label be flipped, in your opinion? **(1 point)**

#BREAKING. #Greece: Molotov cocktails don't fly after protest honouring killed antifa arti... URL via @USER URL

- Provide 2 examples when the model correctly assigned the opposite label after perturbation and 2 examples when the model failed to identify negation. Fill in the table below **(1 point)**

Examples correct	Tweet ID	Original label	Expected label after negation	Model prediction (before negation)	Model prediction (after negation)	Discussion: what is the potential reason for this behavior?
1	16856	1	0	1	0	After negation, the meaning of the sentence changes. (choices -> You are not given any choice). Now the model finds it offensive.
2	96457	1	0	1	0	The offensive content was picked up and negated to a non offensive form. The context of the entire text changes when the first sentence is negated to “don’t take note..). The following statements were offensive originally because it stated “do take note..” at the beginning.
Examples wrong	Tweet ID	Original label	Expected label after negation	Model prediction (before negation)	Model prediction (after negation)	Discussion: what is the potential reason for this behavior?
1	89200	1	0	1	1	Model considers the words “who the hell” as offensive
2	55633	1	0	1	1	Even after negation, other sentences are deemed by the model to be offensive(due to profane

						words like ‘fart’, ‘shit’ and so on)

- How can the model be improved? **(0.5 points)**

The model should be trained on instances and negations. The context of the sentence should be taken into account as automated negation may change the meaning of it. Therefore, negations need to take into account the meaning of sentences in order to provide the best negated form.

Creating examples from scratch with checklist (2.5 points)

7. **Creating negated examples**
   Let us further explore the impact of negations on our offensive language detection model.
   Consider the following templates: ‘*I hate …*’ and ‘*I don’t hate…*’, and fill in the templates below:
   - Use masked language model suggestions: ‘I hate {mask}’ and ‘I don’t hate {mask}’ .
   - Offensive language is often directed towards minority groups. Use the built-in lexicon and explore: ‘I hate {nationality}’, ‘I don’t hate {nationality}’, ‘I hate {religion}’, ‘I don’t hate {religion}’

   Run the model on the created examples.
   - Provide 3 examples when the model assigns the correct label (correct label according to you) and 3 examples when the model fails to assign the correct label (choose both from masking and lexicon suggestions) **(1 point)**

<u>Correct label</u>:
Example 1 (masking):I don't hate it
Example 1 is correctly classified as not offensive(0).

Example 2 (lexicon:nationality): I don’t hate Indian
Example 2 is correctly classified as not offensive (0).

Example 3 (lexicon:religion): I hate Islam
Example 3 is correctly classified as offensive(1).

<u>Incorrect label</u>:
Example 1 (masking): I hate racism
Example 1 is incorrectly classified as offensive(1) whereas it should be not offensive (0).

Example 2 (lexicon:nationality: I hate Saudi
Example 2 is incorrectly classified as not offensive(0) whereas it should be offensive (1).

Example 3 (lexicon:religion): I hate Hinduism
Example 3 is incorrectly classified as not offensive(0) whereas it should be offensive (1).

- Analyze the examples. Can you think of a reason why some examples are classified as offensive while others are not? **(1 point)**

We find that the trained model is overly sensitive to particular keywords that have a standalone negative connotation(Ex Islam, terrorism) and classifies text that contain these keywords as offensive. This is most likely due to systematic biases in the dataset. The model correctly classifies hate speech directed towards certain groups as there are abundant offensive training examples for these groups while misclassifying hateful speech directed towards groups underrepresented in the dataset.

- How can the model be improved? **(0.5 points)**

One solution is to use bias reduction methods and designing data collection processes that account for the social biases. Reweighting the data so that all groups are equally represented. Using methods like subpopulation analysis where a model is evaluated on a target subpopulation in the dataset, this analysis can help in finding out if a model is more biased to a certain group or not.

Individual Task:
The maximum grade you can get for the assignment is an 8. If you want to obtain a better grade, you need to individually submit answers for the task described below on Canvas. If the group project grade is less than an 8, we do not check the individual submission. If the group project grade is an 8 and you submitted an answer for the individual task, you might still only receive an 8, if the quality of the submission is not sufficient.

Task Description:
Develop 2 new diagnostic tests (you can use checklist): describe what they test, explain why they are relevant and implement them. Run the tests and describe your observations. Provide examples of difficult cases, that is, when the model fails to assign the correct label. Discuss potential sources of errors and propose improvements to the model.