**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](https://github.com/idontflow/OLID), which contains 13,240 annotated tweets for offensive language detection. The detailed description of the dataset collection and annotation procedures can be found [here](https://aclanthology.org/N19-1144.pdf). This dataset was used in the SemEval 2019 shared task on offensive language detection ([OffensEval 2019](https://aclanthology.org/S19-2010.pdf)).

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](https://canvas.vu.nl/courses/59974/files/4963294?wrap=1).

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: 41**

**Student 1**

**Name: Bas Willemsen**

**Student id: 2712664**

**Student 2**

**Name: James Bonello**

**Student id: 2717459**

**Student 3**

**Name: Sarah Hassouna**

**Student id: 2601420**

**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** | **67%** | **Amazon is investigating Chinese employees who are selling internal data to third-party sellers looking for an edge in the competitive marketplace. URL #Amazon #MAGA #KAG #CHINA #TCOT** |
| **1** | **4400** | **33%** | **@USER She should ask a few native Americans what their take on this is.** |

1. **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 |
| **1** | **0.27** | **0.48** | **0.34** |
| **0** | **0.71** | **0.50** | **0.59** |
| macro-average | **0.49** | **0.49** | **0.46** |
| weighted average | **0.56** | **0.49** | **0.51** |

|  |  |  |  |
| --- | --- | --- | --- |
| Majority Baseline | | | |
| Class | Precision | Recall | F1 |
| **1** | **0** | **0** | **0** |
| **0** | **0.72** | **1** | **0.84** |
| macro-average | **0.36** | **0.50** | **0.42** |
| weighted average | **0.48** | **0.67** | **0.56** |

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

Run your notebook on [colab](https://colab.research.google.com), 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](https://simpletransformers.ai/): *!pip install simpletransformers*

(you will have to restart your runtime after the installation)

* Follow the [documentation](https://simpletransformers.ai/docs/usage/) 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.

1. 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 |
| **1** | **0.75** | **0.59** | **0.66** |
| **0** | **0.85** | **0.92** | **0.89** |
| macro-average | **0.80** | **0.76** | **0.77** |
| weighted average | **0.82** | **0.81** | **0.81** |

|  |  |  |
| --- | --- | --- |
| Confusion Matrix: Fine-tuned BERT | | |
|  | Predicted Class | |
| Gold Class | **1** | **0** |
| **1** | **141** | **99** |
| **0** | **47** | **573** |

1. Compare your results to the baselines and to the results described in the [paper](https://aclanthology.org/S19-2010.pdf) in 2–4 sentences **(0.5 points)**.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Fine-tuned BERT | | | Majority Baseline | | | Random Baseline | | |
| Class | Precision | Recall | F1 | Precision | Recall | F1 | Precision | Recall | F1 |
| **1** | **0.75** | **0.59** | **0.66** | **0** | **0** | **0** | **0.27** | **0.48** | **0.34** |
| **0** | **0.85** | **0.92** | **0.89** | **0.72** | **1** | **0.84** | **0.71** | **0.50** | **0.59** |
| macro-average | **0.80** | **0.76** | **0.77** | **0.36** | **0.50** | **0.42** | **0.49** | **0.49** | **0.46** |
| weighted average | **0.82** | **0.81** | **0.81** | **0.48** | **0.67** | **0.56** | **0.56** | **0.49** | **0.51** |

**TODO**

1. **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.

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

Number of tokens: **478955**

Number of tokens that have been split into subwords: **91024**

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

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

Average number of subwords per token: **TODO (?)**

1. 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: **EVERYTHING**
2. BERT tokenization: **E ##VE ##R ##Y ##TH ##ING**
3. Morphologically expected split: **EVERY ##THING**
4. 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 characters**

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](https://canvas.vu.nl/courses/59974/files/4963294?wrap=1)) and run the following tests:

**All quantitative results have been rounded to two decimal places.**

1. **Typos** **(6 points)** Spelling variations are sometimes used adversarially to obfuscate and avoid detection ([Vidgen et al., 2019](https://aclanthology.org/W19-3509.pdf); 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.

***Analysis was done with typos=5 since the default setting produced results that are too similar to compare.***

*Quantitative analysis:*

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

***Below are the quantitative results of both datasets by training them on the model of Part A. The precision on the perturbed dataset for both classes is less than for the non-perturbed dataset. This means that the model was more aggressive at mislabeling messages as hate-speech and at mislabeling messages as non-hate-speech when trained on the perturbed data. There is also a noticeable difference in the recall between the datasets on class 1. Clearly, the model trained on the perturbed dataset was less sensitive to detecting hate-speech. Indeed, this leads to a worse F1-score on the perturbed dataset.***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Non-perturbed | | | Perturbed | | |
| Class | Precision | Recall | F1 | Precision | Recall | F1 |
| **1** | **0.90** | **0.70** | **0.79** | **0.89** | **0.62** | **0.73** |
| **0** | **0.75** | **0.92** | **0.83** | **0.71** | **0.92** | **0.80** |
| macro-average |  |  | **0.81** |  |  | **0.76** |

* How many messages were identified correctly in the original dataset, but erroneously after the perturbation? **(1 point)**
  + **81/100 correctly identified in original dataset**
  + **77/100 correctly identified in perturbed dataset**

*Qualitative analysis:*

* Check the add\_typos function in checklist [pertub.py](https://github.com/marcotcr/checklist/blob/master/checklist/perturb.py). How were the typos introduced? **(1 point)**
  + **It does this by using np.random.choice to select a random character(s) and swapping it with its next neighbor.**
* 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)**
  + **A major example would be misspellings such that a neighboring character on the keyboard is accidentally typed in-place of the intended character.**
* Provide 3 examples when the model failed to assign the correct label after perturbation. **(1 point)**

***The examples below are mislabeled as not containing hate-speech by the model trained on the perturbed dataset:***

*Non-perturbed:*

* + ***Row 4:*** *0-1 lost my acca on the first fucking fight cba*
  + ***Row 16:*** *@USER oh fuck off 😂😂*
  + ***Row 29:*** *#Kavanaugh The moment #BlaseyFord is exposed as a fraud, expect a new accuser to surface. Liberals are a one trick pony. #LiberalismIsAMentalDisorder*

*Perturbed:*

* + ***Row 4:*** *0-1 lost my acca on the fisr ftukcing ifght cba*
  + ***Row 16:*** *@USER o hfcuk fof 😂😂*
  + ***Row 29:*** *#Kavanaugh The moment #BlaseyFord is exposed as afraud, expect a new accuser tos ufrace. Liberals are ao ne tric kpony. #LiberalismIsAMentalDisorder*
* What is the main source of the erroneous predictions produced by the model (main source of errors caused by typos)? **(1 point)**
  + ***From the above particular examples, it is interesting to note that the first two out of the three erroneous predictions contain swear words. The perturbation happens on these swear words and it may be the case that the model mislabels messages when it cannot detect swear words.***
* How can the model be improved? **(0.5 points)**
  + ***A larger dataset can definitely by used for perturbation and retrained on that. This way the model can observe more and more varying cases since in this case, the model may have failed to recognize misspelt swear words or insulting phrases as a significant part of the hate-speech.***

1. **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](https://arxiv.org/pdf/2012.15606.pdf); subsection 2.2).

Add negations to the subset and evaluate the model's performance on the perturbed data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Non-perturbed | | | Perturbed | | |
| Class | Precision | Recall | F1 | Precision | Recall | F1 |
| **1** | **0.90** | **0.70** | **0.79** | **0.89** | **0.66** | **0.76** |
| **0** | **0.75** | **0.92** | **0.83** | **0.73** | **0.92** | **0.81** |
| macro-average |  |  | **0.81** |  |  | **0.79** |

* ***Both models seem to have a similar performance. If any, the perturbed dataset seems to perform slightly worse than the original dataset, but the difference can be called insignificant.***

*Qualitative analysis:*

* Check the add\_negation function in checklist [pertub.py](https://github.com/marcotcr/checklist/blob/master/checklist/perturb.py). What kind of negations does it produce? **(1 point)**
  + ***It checks for the root word and skips unless it is a verb or auxiliary word. Then, it checks the lemma of the word and applies the negation accordingly.***

***Please refer to the subset\_negs\_test\_df.html file on the code submission for this analysis.***

* Look at the created negated sentences, are they linguistically correct? Provide 2–5 examples of linguistically incorrect sentences. **(1 point)**
  + ***Row 26:*** *@USER @USER @USER @USER Beats the hell NJ out of me lol. Dude* ***already didn't make*** *the assumption that I'm antifa. He even ignored all the evidence to make that assumption.*
  + ***Row 27:*** *#ConfirmKavanugh now,* ***stall tactics are not DC cronies only recourse****. No more delays. #VoteRed to end this madness and to #MAGA*
  + ***Row 66:*** *#antifa #Resist.. Trump* ***is not trying to bring world peace, not obstruct*** *like the democrats.. is this good for you also? or haters gonna hate? URL*
* Check the first 10 negated messages. For which of these negated messages should the label be flipped, in your opinion? **(1 point)**
  + ***Row 0, Row 2, and Row 6 should be flipped since they do not contain any profanity or directed hate/accusations towards a particular person or people.***
* 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 | Discussion: what is the potential reason for this behavior? |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| Examples wrong | Tweet ID | Original label | Expected label after negation | Model prediction | Discussion: what is the potential reason for this behavior? |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |

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

***The model can be improved by finetuning where the negation should occur. For example, a negation can occur at random rather than at every possible location. This may increase reduce nonsense such as double negatives as seen above. If this increases the effectiveness of the dataset, then it will improve model training.***

**Creating examples from scratch with checklist (2.5 points)**

1. **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)**
  + ***When the model assigns the correct label:***
    - **Mask suggestions**
      * **Row 20:** I hate men. **Prediction:** 1
      * **Row 22:** I don't hate racism . **Prediction:** 1
      * **Row 27:** I hate technology. **Prediction:** 0
    - **Lexicon suggestions**
      * **Row 78:** I hate Tunisian. **Prediction:** 1
      * **Row 83:** I hate Agnosticism. **Prediction:** 0
      * **Row 20:** I don't hate Sikhism. **Prediction:** 0
  + ***When the model fails to assign the correct label:***
    - **Mask suggestions**
      * **Row 16:** I hate hate **Prediction:** 1
      * **Row 2:** I hate capitalism **Prediction:** 1
      * **Row 11:** I hate this **Prediction:** 1
    - **Lexicon suggestions**
      * **Row 66:** I hate South Sudanese. **Prediction:** 0
      * **Row 84:** I hate Islam. **Prediction:** 1
      * **Row 68:** I hate Slovenian **Prediction:** 0
* Analyze the examples. Can you think of a reason why some examples are classified as offensive while others are not? **(1 point)**

***While this may be a subjective matter, the phrase* ‘*I hate Islam*’ *is essentially classified as offensive due to the common appearance of Islamophobia online. However, hate towards a religion may not necessarily mean that it represents hate towards people who follow that religion. Now, as a word, ‘racism’ represents hate towards a race. If a user doesn’t hate something that represents hate, then that means that this user condones hate towards a race, and essentially ‘I don’t hate racism’ correctly translates as hate-speech. It seems that the more politically related a term is, the more sensitive the model may be to classifying the surrounding message as hate speech. In fact, the model incorrectly classifies the hatred of capitalism as hate-speech. On the other hand, it seems that inanimate objects such as technology, or robots are less likely to be related to hate-speech according to the model, while hating on almost all living-beings would translate as hate-speech for the model.***

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

***Again, ideally, the model should be made more robust to differentiate hatred towards a people, or a race, and hatred towards a concept. Hating racism relates to the hatred of prejudice towards a particular people while hating Islam relates to the hatred of a religion but not necessarily the people who follow it. More diverse examples may help the model to be more sensitive to detect prejudice towards a people rather than prejudice towards a concept.***

***Please note that all the dataframes of results used for part B can be found as part of the code submission as .html files. All quantitative and qualitative analysis was done using these same exact dataframes.***

**BONUS:**

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.