

ADVANCE Labs - Cyberbullying Detection Lab

Copyright © 2021 - 2023.

The development of this document is partially funded by the National Science Foundation's Security and Trustworthy Cyberspace Education, (SaTC-EDU) program under Award No. 2114920. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation. A copy of the license can be found at <http://www.gnu.org/licenses/fdl.html>.

1 Lab Overview

In this lab, you will learn about how AI/ML can be used to detect societal issues such as cyberbullying. Cyberbullying is bullying performed via electronic means such as mobile/cell phones or the Internet. The objective of this lab is for students to gain practical insights into online harassment such as cyberbullying, and to learn how to develop AI/ML solutions to defend against this problem.

In this lab, students will be given a starter-code. Their task is to follow the instructions provided in the Jupyter notebook, train an AI/ML model on the given dataset, evaluate their model, and deploy the model by testing it on their own samples. In addition to the attacks, students will also be guided to perform hyperparameter tuning to further improve the performance of their detection models. Students will be asked to evaluate whether their tuning effort improves their detection models or not. This lab covers the following topics:

- Detection of cyberbullying in tweets
- Hyperparameter tuning to improve model performance

Disclaimer: This lab and AI models are intended for education and research purposes only. The lab could contain potentially triggering language and deal with difficult subject material. We have minimized showing such language samples in this lab. They do not represent the views of the authors.

2 Lab Environment

This ADVANCE lab has been designed as a [Jupyter notebook](#). ADVANCE labs have been tested on the [Google Colab platform](#). We suggest you to use Google Colab, since it has nearly all software packages preinstalled, is free to use and provides free GPUs. You can also download the Jupyter notebook from the lab website, and run it on your own machine, in which case you will need to install the software packages yourself (you can find the list of packages on the ADVANCE website). However, most of the ADVANCE labs can be conducted on the cloud, and you can follow our instructions to create the lab environment on the cloud.

3 Lab Tasks

3.1 Getting Familiar with Jupyter Notebook

The main objective of this lab is to learn how AI/ML can be used to detect online harassment, such as cyberbullying. Before proceeding to that, let us get familiar with the Jupyter notebook environment.

Jupyter notebooks have a Text area and a Code area. The Text area is where you'll find instructions and notes about the lab tasks. The Code area is where you'll write and run code. Packages are installed using `pip`, and need to be preceded with a `!` symbol. Try accessing the lab environment for this task [here](#).

The lab has three areas: one text area and two code areas. Follow the instructions for the three areas, fill the three areas with the instructed content and add a screenshot to your report.

3.2 Cyberbullying Detection

In this lab, you will develop AI to detect cyberbullying. You will use a dataset of tweets to train your AI model, evaluate the performance of your AI model, and then deploy it by running it against your own samples. You can access the lab by clicking [here](#).

3.2.1 Datasets Selection

In this lab, we provide three datasets: formspring dataset, Davidson dataset and Founta dataset. You can edit the name of dataset in following code:

```
main_df = pd.read_csv('CyberbullyingLab1/formspring_dataset.csv', sep = '\t')
```

In this lab, you will be using the formspring dataset, a widely used dataset of cyberbullying texts. Run all the code until this part of the lab. Report the size of the training, testing and validation sets.

3.2.2 Preprocessing Data

Follow the instructions in the text areas and run the subsequent codes to preprocess your data, as follows. Here is a sample from the lab:

```
spacy_en = spacy.load('en_core_web_sm')
text = spacy_en("the cat sat on the mat.")
```

Add code to preprocess the following cyberbullying text, and include the generated tokens in your report: "Harlem shake is just an excuse to go full retard for 30 seconds". You can add a code block to run your code.

Proceed to further preprocess your dataset by using pretrained word embeddings.

3.2.3 Model Training

After you have preprocessed the dataset, the next task is to train the AI model. Follow the lab instruction to train the AI model. What is the final training accuracy that your model achieves?

```
print(f'Train Acc: {____*100:.2f}%')
```

3.2.4 Model Evaluation

Now it is time to run your trained model on a test dataset. Recall that we have already partitioned the dataset into train, validation and test sets. Run your model on the test partition and report your results here. Use the evaluate function.

```
____, ____ = evaluate(____) # complete this code
print(f'| Test Loss: {____:.3f} | Test Acc: {____*100:.2f}% |')
```

3.2.5 Deploy and Run Custom Samples

Copy the sentences from the [samples](#) file and use your model to check if they contain cyberbullying content. Report the samples that were detected as cyberbullying. Do you think your model is good enough? In this lab, you will later learn how to use hyperparameters to improve the performance of your model.

3.3 Hyperparameter Tuning for Cyberbullying and Hate-speech Detection

In the rest tasks, various hyperparameters will be tuned to observe the effect on the model. Hyperparameters determine the parameters (weight and bias) of your AI model. Hyperparameters include learning rate α , number of epochs, number of hidden layers/dimensions, number of hidden units, choice of activation function, mini-batch size, etc. In this task, we will focus on the learning rate, the number of epochs, and the number of hidden dimensions.

The tasks are to be completed in [hyperparameter_tuning.ipynb](#) which contains different sections from 0 through 7. Run all the cells in sections 0 through 6 to download and load the required files and modules and to initialize the required variables and functions. In section 7 (Hyperparameter tuning), run the first cell to download the embedding and to build our vocabulary. The second cell in section 7 contains the hyperparameters to be tuned. Complete the following tasks:

3.3.1 Train Model Using Default Hyperparameters

Execute the second cell of section 7, with the following configurations:

```
embedding_dim = 100
hidden_dim = 374
output_dim = 2
number_of_epochs = 15
learning_rate = 0.001
```

Train the model using these configurations by running the third cell. Running the third cell outputs the training information as the model trains, computes the final test accuracy, and plots a graph of training/validation loss against the number of epochs. Write your observations in your report, including screenshots of the outputs.

3.3.2 Hidden Dimension

The input text sequence (or sentence) is converted (tokenized) into individual words. Each word in the sentence is converted into a vector representation using for example a one-hot vector (see Fig. 1) or embedding vector (see 3.3.4 and Fig. 2). Each vector representation of the words in the sentence is fed into the hidden layer of the long short-term memory (LSTM) RNN to produce an output, this output serves as input to the hidden layer together with the vector representation of the next word and so on, until we arrive at the last word in the sentence. The hidden dimension is therefore the size of the number of units contained in the hidden layer. If the hidden layer has 374 units and our word vector is 100 dimensional then a weight matrix of size 374 x 100 is produced. Note that the one-hot vectors are sparse meaning it contains a lot of zeros and processing time will be affected as the vocabulary size grows. Also, they don't have embedded meaning and do not take into account the order of words in a text as opposed to word embedding.

In this task, the effect of varying the hidden dimension size is observed. For each of the hidden dimensions (150, 500, and 650), set the other hyperparameters as they were in section 3.3.1 and train the models. Compare the outputs of the different models and report your observations.

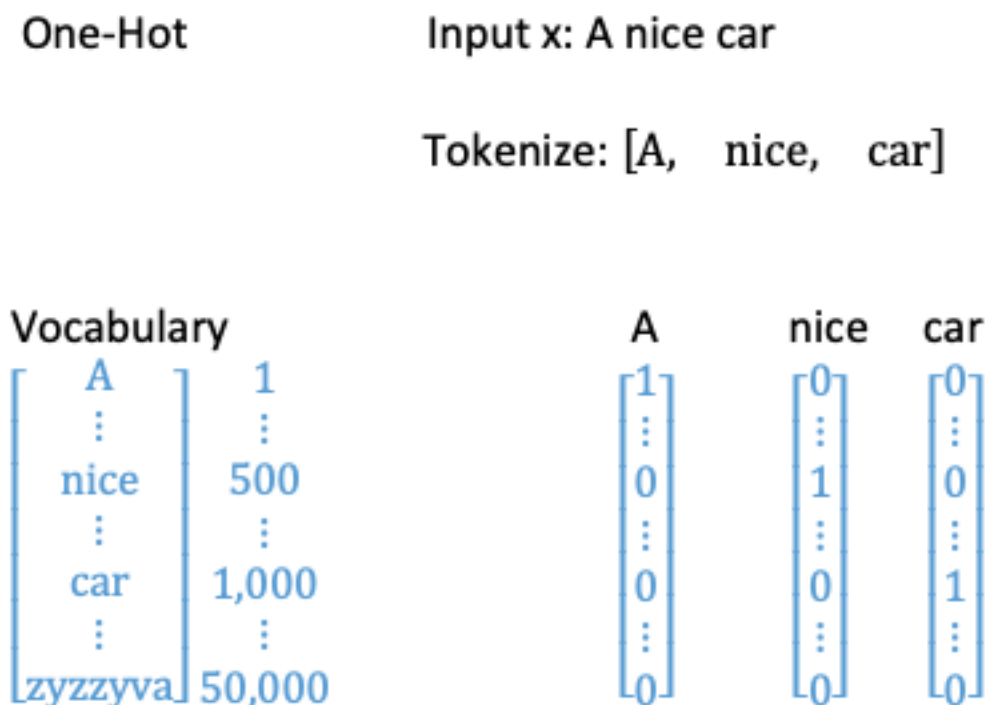


Figure 1: One-hot vector

3.3.3 Number of Epochs

Training AI models requires going through our dataset multiple times to learn the optimal parameters. Number of epochs is the number of times we go through our entire dataset in order to improve performance. In this task, we will observe how varying the number of epochs affects the model.

For each of the hidden dimensions (500 and 650), set the other hyperparameters as they were in section 3.3.1 except for `hidden_dim` and `number_of_epochs`. For each dimension, set the `number_of_epochs` to 25 and 50, and train the model. What are your observations? How did these changes affect the accuracy and loss?

3.3.4 Embedding Dimension

A word embedding is a representation of words in high dimensional space, so a word like *girl* can be represented as a 100-dimensional vector: [4.987, -0.1011, ..., 0.16728]. Word embeddings are vectors that carry meaning (e.g. semantic distance) and can be trained using neural networks, GloVe and Word2Vec as examples of pre-trained word embeddings. When trained, they learn similarities between vectors. If we have the embedding vectors of *man*, *woman* and *boy* as e_{girl} , e_{man} , and e_{boy} . We can answer the question: a boy is to a girl as a man is to ? by taking the word that its embedding produces the maximum cosine similarity by computing


$$f(e_{girl} - e_{boy}, e_{word_i} - e_{man})$$

Where e_{word_i} is the embedding of each word i in our vocabulary and $f(.)$ is the cosine similarity given by

$$CosineSimilarity(a, b) = \frac{a \cdot b}{\|a\|_2 \|b\|_2}$$

$a \cdot b$ is the dot product of vectors a and b , and $\|a\|_2$ is the norm or length of the vector a . Embeddings can have different dimensions, in this task, we will be using the GloVe embedding - the 200 dimensional Twitter embedding to train our AI model.

Embedding



A	nice	car
0.000123	0.000789	0.590295	0.201110	-0.008745
⋮	⋮	⋮	⋮	⋮
0.000400	-0.087653	0.925957	-0.109382	0.090183
⋮	⋮	⋮	⋮	⋮
-0.198465	0.2134671	0.827125	0.650941	-0.876341

Figure 2: 100 dimensional Word embeddings

As the first cell of section 7 Task 8, we change the embedding to the Twitter embedding (`pretrained_twitter_embedding()`) and execute the cell to download the embedding and build our vocabulary. Set the hyperparameters in cell two as in section 3.3.1 except for the embedding dimension which should be set to 200. Train the model by running cell three, and compare the results to the results of default hyperparameters.

3.3.5 Learning Rate

Learning rate needs to be chosen carefully in order for gradient descent to work properly. How quickly we update the parameters of our models is determined by the learning rate. If we choose the learning rate to be too small, we may need a lot more iteration to converge to the optimal values. If we choose the learning rate to be too big, we may go past our optimal values. So, it is important to choose the learning rate carefully.

To complete this task, re-run the first cell of section 7 to change the embedding to the default embedding (`pretrained_default_embedding()`) and run the cells under **Task 9: Compare different learning rates**. The hyperparameters in the first cell have been set to the default values. Cell 2 compares different learning rate values (0.9, 0.1, 0.05, and 0.001). A model is trained for each of the learning rate values and the metrics stored. For each learning rate, the training loss is plotted against the number of epochs. Observe the outputs and report your observations.

3.3.6 Output Dimension

Change the output dimension (`output_dim`) to any other integer value and run the next cell. Why does this fail?

3.3.7 Discussion

We experimented with different hyperparameters in this lab. What can you conclude about training AI models? Specifically, what are your observations about the model before Vs. after hyperparameter tuning?

4 Submission Instructions

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to explain the observations that are interesting or surprising. Please also list important code snippets followed by explanations. Simply attaching code without any explanation will not receive credits.