Hate_Speech_Detection_Dynabench.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

Share Connect Editing

Hate Speech Detection on Dynabench

Open Tweet Share Notebook NEW

Notebook to train an RoBERTa model to perform hate speech detection. The dataset used is the Dynabench Task - Dynamically Generated Hate Speech Dataset from the paper Vidgen et al. (2020).

> The dataset provides 40,623 examples with annotations for fine-grained labels, including a large number of challenging contrastive perturbation examples. Unusually for an abusive content dataset, it comprises 54% hateful and 46% not hateful entries.

There is no published state-of-the-art model on this dataset at this point though the task on Dynabench reports mean MER (Model Error Rate) scores. We are able to train the model to a F1 score of **86.6%** after only 1 epoch.

The notebook is structurated as follows:

- Setting up the GPU Environment
- Getting Data
- Training and Testing the Model
- Using the Model (Running Inference)

Task Description

Hate Speech Detection is the automated task of detecting if a piece of text contains hate speech.

Setting up the GPU Environment

Ensure we have a GPU runtime

If you're running this notebook in Google Colab, select `runtime` > `Change Runtime type` from the menubar. Ensure that `GPU` is selected as the `Hardware accelerator`. This will allow us to use the GPU to train the model subsequently.

Install Dependencies and Restart Runtime

```
[ ] !pip install -q transformers
    !pip install -q simpletransformers
```

```
|▏                                | 24kB 13.39M/s
|▏                                | 31kB 10.90M/s
|▏                                | 1.8MB 17.5M/s
|▏                                | 7.4MB 38.9M/s
|▏                                | 1.3MB 17.6M/s
|▏                                | 51kB 5.8M/s
|▏                                | 71kB 10.3M/s
|▏                                | 133kB 43.4M/s
|▏                                | 81kB 11.4M/s
|▏                                | 245kB 53.7M/s
|▏                                | 163kB 51.4M/s
|▏                                | 4.5MB 41.8M/s
|▏                                | 112kB 57.2M/s
|▏                                | 1.1MB 50.2M/s
|▏                                | 61kB 8.4M/s
  Building wheel for sacrelf (setup.py) ... done
  Building wheel for subprocess32 (setup.py) ... done
  Building wheel for blinker (setup.py) ... done
ERROR: google-colab 1.0.0 has requirement ipykernel~=4.10, but you'll have ipykernel 5.4.2 which is incompatible.
ERROR: datasets 1.2.0 has requirement tqdm<4.50.0,>=4.27, but you'll have tqdm 4.55.1 which is incompatible.
```

You might see the error `ERROR: google-colab X.X.X has requirement ipykernel~=X.X, but you'll have ipykernel X.X.X which is incompatible` including the dependencies. **This is normal** and can be safely ignored as the `simpletransformers` library.

The **solution** to this will be to **reset the execution environment** now. Go to the menu `Runtime` > `Restart runtime` then continue on from the next section to download and process the data.

Getting Data

Pulling the data from Github

The code below uses `pandas` to pull the dataset as a CSV file from the official Github repository. The dataset is now stored as a dataframe, in which we can transform for use with the `simpletransformers` library to train the model. So we pull the CSV from Github and split the CSV into training and test sets by using the column `split` in the CSV file which indicates which example/sample is a training sample and which is a test sample.

```
[ ] import pandas as pd

    df = pd.read_csv('https://raw.githubusercontent.com/bidgen/Dynamically-Generated-Hate-Speech-Dataset/main/2020-11-3-DynamicallyGeneratedHateDataset-e
    train_df = df[df['split'] == 'train'] # split the dataset by the column 'split' which labels 'train' and 'test' samples
    test_df = df[df['split'] == 'test'] # split the dataset by the column 'split' which labels 'train' and 'test' samples
```

Once done we can take a look at the `head()` of the training set to check if our data has been retrieved properly.

```
[ ] train_df.head()
```

	Unnamed: 0	id	text	label	type	model_wrong	db.model.preds	pred_probs	status	round	split	annotator
0	1	10000	It should be legal for a husband to rape his wife	hate	notgiven	True	0.974567	dynabench.entry	1	train	gruzjuwe3	
2	3	10002	dalits are lowlives	hate	notgiven	True	0.92319	dynabench.entry	1	train	cagrizzlyen	
4	5	10004	I don't work this hard so that those immigrant...	hate	notgiven	True	0.98836	dynabench.entry	1	train	cagrizzlyen	
5	6	10005	I don't work this hard so that those migrants ...	hate	notgiven	True	0.99506	dynabench.entry	1	train	cagrizzlyen	
6	7	10006	I don't work this hard so that those foreigner...	hate	notgiven	True	0.99340	dynabench.entry	1	train	cagrizzlyen	

We transform the dataframe column `label` so that the labels `hate` and `nothate` are now integers : and 0 respectively. This input format of labels is required for our training step with the `transformers` library.

```
[ ] train_df = train_df.replace({'label': {'hate': 1, 'nothate': 0}}) # relabel the 'label' column, hate is 1 and nothate is 0
    test_df = test_df.replace({'label': {'hate': 1, 'nothate': 0}}) # relabel the 'label' column, hate is 1 and nothate is 0
    train_df.head()
```

	Unnamed: 0	id	text	label	type	model_wrong	db.model.preds	pred_probs	status	round	split	annotator
0	1	10000	It should be legal for a husband to rape his wife	1	notgiven	True	0.974567	dynabench.entry	1	train	gruzjuwe3	
2	3	10002	dalits are lowlives	1	notgiven	True	0.92319	dynabench.entry	1	train	cagrizzlyen	
4	5	10004	I don't work this hard so that those immigrant...	1	notgiven	True	0.98836	dynabench.entry	1	train	cagrizzlyen	
5	6	10005	I don't work this hard so that those migrants ...	1	notgiven	True	0.99506	dynabench.entry	1	train	cagrizzlyen	
6	7	10006	I don't work this hard so that those foreigner...	1	notgiven	True	0.99340	dynabench.entry	1	train	cagrizzlyen	

We also rename the `label` column to `labels` as this also conforms to the input format required for the `simpletransformers` library.

```
[ ] train_df = train_df.rename(columns={'label': 'labels'})
    test_df = test_df.rename(columns={'label': 'labels'})
```

We can now take a look at the train and test set sizes. We see that this dataset is quite special as the `hate` and `nothate` class sizes are actually quite close in proportion.

```
[ ] data = [[train_df.labels.value_counts()[0], test_df.labels.value_counts()[0]],
            [train_df.labels.value_counts()[1], test_df.labels.value_counts()[1]]]
    # Prints out the dataset sizes of training text and validation sets per the labels.
    pd.DataFrame(data, columns=['Train', 'Test'])
```

	Train	Test
0	14013	1857
1	17884	2205

Training and Testing the Model

Set the Hyperparameters

First we setup the hyperparameters. We train for 1 epoch only as we want the training to complete fast. The important parameters are the `max_seq_length`, which we set to 64 and `sliding_window` to true. As we don't have a high RAM GPU on Colab we can't set the `max_seq_length` to be large in value and by using `sliding_window` we'll least are able to handle longer sequences without truncation. See the `simpletransformers` documentation for a more detailed explanation of a sliding window.

```
[ ] train_args = {
        'reprocess_input_data': True,
        'overwrite_output_dir': True,
        'sliding_window': True,
        'max_seq_length': 64,
        'num_train_epochs': 1,
        'train_batch_size': 128,
        'fp16': True,
        'output_dir': '/outputs/',
    }
```

Train the Model

Once we have setup the hyperparameters in the `train_args` dictionary, the next step would be to train the model. We use the RoBERTa model from the awesome Hugging Face Transformers library and use the Simple Transformers library on top of it to make it so we can train the classification model with just 2 lines of code.

RoBERTa is an optimized BERT model by Facebook Research with better performance on the masked language modeling objective that modifies key hyperparameters in BERT, including removing BERT's next-sentence pretraining objective, and training with much larger mini-batches and learning rates. In short, its a bigger but generally better performing BERT model we can easily plug in here with the transformers library.

```
[ ] from simpletransformers.classification import ClassificationModel
    import pandas as pd
    import logging
    import sklearn

    logging.basicConfig(level=logging.DEBUG)
    transformers_logger = logging.getLogger('transformers')
    transformers_logger.setLevel(logging.WARNING)

    # We use the XLMnet base cased pre-trained model.
    model = ClassificationModel('roberta', 'roberta-base', num_labels=2, args=train_args)

    # Train the model, there is no development or validation set for this dataset
    # https://huggingface.co/docs/transformers/model_doc/roberta#roberta
    model.train_model(train_df)

    # Evaluate the model in terms of accuracy score
    result, model_outputs, wrong_predictions = model.eval_model(test_df, acc=sklearn.metrics.accuracy_score)
```

```
DEBUG:filelock:Attempting to release lock 140363797556776 on /root/.cache/huggingface/transformers/7316a6e13e5f6e99e6531281d51189996b2f7b8bd7e08c7b
INFO:filelock:Lock 140363797556776 released on /root/.cache/huggingface/transformers/7316a6e13e5f6e99e6531281d51189996b2f7b8bd7e08c7b8b8f5b4a31b
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): huggingface.co:443
DEBUG:urllib3.connectpool:https://huggingface.co:443 "HEAD /roberta-base/resolve/main/pytorch_model.bin HTTP/1.1" 302 0
DEBUG:filelock:Attempting to acquire lock 140363798017264 on /root/.cache/huggingface/transformers/5334a66ff7ff9a7cdfa95d1a8561677371138708ba7271
INFO:filelock:Lock 140363798017264 acquired on /root/.cache/huggingface/transformers/5334a66ff7ff9a7cdfa95d1a8561677371138708ba7271b2c8b8d881aa4
DEBUG:urllib3.connectpool:Starting new HTTPS connection (1): cdn-lfs.huggingface.co:443
DEBUG:urllib3.connectpool:https://cdn-lfs.huggingface.co:443 "GET /roberta-base/7f8fe2e9739c43929ee9b81865343d4e280e1d8931ef37ea839e1e1b9e463b 5
Downloading: 100%|██████████|  501M/501M [00:00:00, 72.8MB/s]
DEBUG:filelock:Attempting to release lock 140363798017264 on /root/.cache/huggingface/transformers/5334a66ff7ff9a7cdfa95d1a8561677371138708ba7271b2c
INFO:filelock:Lock 140363798017264 released on /root/.cache/huggingface/transformers/5334a66ff7ff9a7cdfa95d1a8561677371138708ba7271b2c8b8d881aa4.
Some weights of the model checkpoint at roberta-base were not used when initializing RobertaForSequenceClassification: ['lm_head.bias', 'lm_head.dense
- This is expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model trained on another task or with another arc
- This is NOT expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model that you expect to be exactly identical
Some weights of RobertaForSequenceClassification were not initialized from the model checkpoint at roberta-base and are newly initialized: ['classifier
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
DEBUG:urllib3.connectpool:Starting new HTTPS connection (1): huggingface.co:443
DEBUG:urllib3.connectpool:https://huggingface.co:443 "HEAD /roberta-base/resolve/main/vocab.json HTTP/1.1" 200 0
DEBUG:filelock:Attempting to acquire lock 140363797904584 on /root/.cache/huggingface/transformers/d5ccdbfe6baa747ef204324697fc32ea3fa6563b37b0e253
INFO:filelock:Lock 140363797904584 acquired on /root/.cache/huggingface/transformers/d5ccdbfe6baa747ef204324697fc32ea3fa6563b37b0e253cfce2b01f6f3.
DEBUG:urllib3.connectpool:Starting new HTTPS connection (1): huggingface.co:443
DEBUG:urllib3.connectpool:https://huggingface.co:443 "GET /roberta-base/resolve/main/vocab.json HTTP/1.1" 200 898823
Downloading: 100%|██████████|  899k/899k [00:00:00, 12.6MB/s]
DEBUG:filelock:Attempting to release lock 140363797904584 on /root/.cache/huggingface/transformers/d5ccdbfe6baa747ef204324697fc32ea3fa6563b37b0e253
INFO:filelock:Lock 140363797904584 released on /root/.cache/huggingface/transformers/d5ccdbfe6baa747ef204324697fc32ea3fa6563b37b0e253cfce2b01f6f3.
DEBUG:urllib3.connectpool:Starting new HTTPS connection (1): huggingface.co:443
DEBUG:urllib3.connectpool:https://huggingface.co:443 "HEAD /roberta-base/resolve/main/merges.txt HTTP/1.1" 200 0
DEBUG:filelock:Attempting to acquire lock 140363797904584 on /root/.cache/huggingface/transformers/cafdec90fc6017061a1ec815d57040cb9fea39da6d0178
INFO:filelock:Lock 140363797904584 acquired on /root/.cache/huggingface/transformers/cafdec90fc6017061a1ec815d57040cb9fea39da6d0178119ef0f82f7ff.
DEBUG:urllib3.connectpool:Starting new HTTPS connection (1): huggingface.co:443
DEBUG:urllib3.connectpool:https://huggingface.co:443 "GET /roberta-base/resolve/main/merges.txt HTTP/1.1" 200 456518
Downloading: 100%|██████████|  456k/456k [00:00:00, 11.5MB/s]
DEBUG:filelock:Attempting to release lock 140363797904584 on /root/.cache/huggingface/transformers/cafdec90fc6017061a1ec815d57040cb9fea39da6d0178
INFO:filelock:Lock 140363797904584 released on /root/.cache/huggingface/transformers/cafdec90fc6017061a1ec815d57040cb9fea39da6d0178119ef0f82f7ff.
INFO:simpletransformers.classification.classification_model: 15576 features created from 32497 samples.
Epoch 1 of 1: 100%|██████████|  1/1 [12:10<00:00, 730.19s/it]
Epochs 0/1. Running Loss: 0.5345: 100%|██████████|  277/277 [02:10<00:00, 2.47it/s]
/usr/local/lib/python3.6/dist-packages/torch/optim/lr_scheduler.py:216: UserWarning: Please also save or load the state of the optimizer when saving o
  warnings.warn(SAVE_STATE_WARNING, UserWarning)
INFO:simpletransformers.classification.classification_model: Training of roberta model complete. Saved to /outputs/.
INFO:simpletransformers.classification.classification_model: Converting to features started. Cache is not used.
INFO:simpletransformers.classification.classification_model: Sliding window enabled
100%|██████████|  4862/4862 [00:42<00:00, 1639.83it/s]
INFO:simpletransformers.classification.classification_model: 4862 features created from 4862 samples.
Running Evaluation: 100%|██████████|  152/152 [00:50<00:00, 49.36it/s]
INFO:simpletransformers.classification.classification_model: {'acc': 0.7595334969374414, 'tp': 1851, 'tn': 1639, 'fp': 218, 'fn': 354, 'acc': 0.859142
```

The model has been trained and evaluating on the test set after training to only 1 epoch gives an accuracy of **85.9%**. We want to also evaluate the F1 score which is a better measure as the dataset is slightly imbalanced.

```
[ ] result, model_outputs, wrong_predictions = model.eval_model(test_df, acc=sklearn.metrics.f1_score)
```

```
INFO:simpletransformers.classification.classification_model: Converting to features started. Cache is not used.
INFO:simpletransformers.classification.classification_model: Sliding window enabled
100%|██████████|  4862/4862 [00:42<00:00, 1779.92it/s]
INFO:simpletransformers.classification.classification_model: 4862 features created from 4862 samples.
Running Evaluation: 100%|██████████|  152/152 [00:50<00:00, 57.12it/s]
INFO:simpletransformers.classification.classification_model: {'acc': 0.7595334969374414, 'tp': 1851, 'tn': 1639, 'fp': 218, 'fn': 354, 'acc': 0.866167524
```

We see that the output F1 score from the model after training for 1 epoch is **86.6%** (acc: 0.866167524671503).

Using the Model (Running Inference)

Running the model to do some predictions/inference as simple as calling `model.predict([input_list])`

```
[ ] samples = [test_df[test_df['labels'] == 0].sample(1).iloc[0]['text']] # get a random sample from the test set which is nothate
    predictions, _ = model.predict(samples)
    label_dict = {0: 'nothate', 1: 'hate'}
    for idx, sample in enumerate(samples):
        print('{: <{}} -> {}'.format(sample, len(max(label_dict[predictions[idx]], sample))
```

```
INFO:simpletransformers.classification.classification_model: Converting to features started. Cache is not used.
INFO:simpletransformers.classification.classification_model: Sliding window enabled
100%|██████████|  1/1 [00:00<00:00, 5.22ms/it]
INFO:simpletransformers.classification.classification_model: 1 features created from 1 samples.
100%|██████████|  1/1 [00:0<00:00, 14.48ms/it]
0 - nothate: don't call this place of garbage a 70
```

We can also generate a `results.txt` file from the test set. The file is stored in our Colab environment. Hit the `folder` icon at the side and you can download the `results.txt` file from the file browser. You can submit this `.txt` file to Dynabench for evaluation if you wish to.

```
[ ] predictions, _ = model.predict(test_df['text'].tolist())
    df = pd.DataFrame(predictions)
    df.to_csv('results.txt', index=False, header=False) # saves the prediction results to a file in the colab environment
```

```
INFO:simpletransformers.classification.classification_model: Converting to features started. Cache is not used.
INFO:simpletransformers.classification.classification_model: Sliding window enabled
100%|██████████|  4862/4862 [00:42<00:00, 1785.92it/s]
INFO:simpletransformers.classification.classification_model: 4862 features created from 4862 samples.
100%|██████████|  152/152 [00:50<00:00, 54.67it/s]
```

We can connect to Google Drive with the following code to save any files you want to persist. You can also click the `files` icon on the left panel and click `Mount Drive` to mount your Google Drive.

The root of your Google Drive will be mounted to `/content/gdrive/My Drive/`. If you have problems mounting the drive, you can check out this tutorial.

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

You can move the model checkpoint files which are saved in the `/content/outputs/best_model/` directory to your Google Drive.

```
[ ] import shutil
    shutil.move('/outputs/', '/content/drive/My Drive/outputs/')
```

More Notebooks @ wuyenesuyen-practical-ml and do drop us some feedback on how to improve the notebooks on the Github repo.

Colab paid products - Cancel contracts here