[F23] PracticalMLandDL

Final Technical Report

---

1. **Roman Makarov**
   o.makarov@innopolis.university
2. **Adela Krylova**
   a.krylova@innopolis.university

Topic:

Disaster tweet Generation and Detection using GAN-like structure

---

GitHub: https://github.com/rmakarovv/pmldl_project

What we tried:

Generator:

1. **Training the generator on the whole corpus data**
   We gave the generator both real and fake tweets, so that it learnt to distinguish between them. The training data was formatted like:
   [real/fake label]>>>[text of the tweet].
   This approach was unstable due to the fact that sometimes for a prompt to generate a real-like tweet, GPT generated samples that come from fake data, hence the generating capabilities of it have been significantly decreased.
2. **Training the generator only on positive data**
   We tried to train the gpt2 model only on real tweets without showing it any fake examples. This approach brought fake tweets of a good quality and with the further fine-tuning we were able to achieve remarkable results.

Discriminator:

1. **Training BERT model from tensorflow**
   Although this approach gave stable and good convergence, during our project completion time one of the libraries was changed, so that the total solution became incompatible
2. **Training small BERT with keras_nlp**
   For pre-training the model before the fine-tuning stage alongside the Generator model, we utilized 200 samples of real

and fake data. At this stage the model was able to correctly predict **57% labels**.

Evaluator:

1. **DistilBERT**
   To obtain the evaluator model we initialized DistilBERT with 66 million parameters and trained it on the whole corpus of available data.

Final solution:

We decided to attempt this task with a novel approach which promises a more general classifier and enhanced generator. For that purpose we used a GAN-like architecture. We utilized GPT2 pre-trained on 500-1000 samples from the training data provided on Kaggle as a generator and a small BERT model pre-trained on 200 samples from the training data as a discriminator.

After pre-training stage, fine-tuning goes in the following way:

1. Generator produces N fake tweets.
2. N real tweets are taken from the dataset and mixed with fake ones from the previous stage.
3. Discriminator is tested and trained on this data.
4. Generator is trained further on real data or on the fake tweets that were able to fool the discriminator.

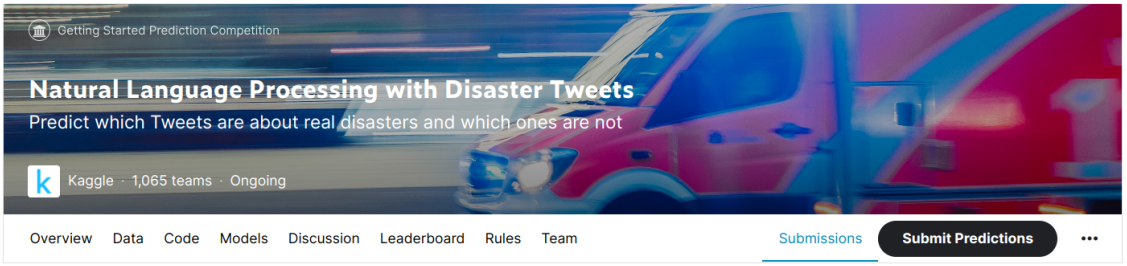
After fine-tuning models are evaluated in the following way:

1. Discriminator's predictions before and after fine-tuning are submitted to kaggle competition.
2. N generated tweets after fine-tuning are given to the evaluator model, and its accuracy is measured.

Results

1. DistilBERT (evaluator) was trained on **10000 samples** and achieved **0.84** accuracy.
2. Discriminator's accuracy was **improved from 0.57 to 0.68** (**20%** increase) after fine-tuning. The model was only trained on **300 real and 100 generated** samples, utilizing only **3%** of available data

3. The Generator model was able to fool Evaluator in **72%** of its generated samples. The evaluator model's accuracy was dropped from **83.5%** (public score on kaggle) to **28%**, which is a decrease of **3** times.




```
[13]: X_val = [x.strip() for x in open('generated.txt', 'r').readlines()]
      y_val = [LABEL_FAKE for i in range(len(X_val))]

      classifier.evaluate(X_val, y_val)

      2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 4s/step - accuracy: 0.2775 - loss: 2.1003
[13…  [2.1539533138275146, 0.25999999046325684]
```

Timeline of the project:

Our project timeline was divided into 4 runs - the results of each part were presented in the intermediate reports.

➔ **1st run:**

We analyzed the dataset, and figured out what we could do with it, so that it would bring something new. We revised GANs technology.

➔ **2nd run:**

We wrote two separate models. We discovered several approaches on doing the generator and discriminator. First drafts of generators based on gpt2 and discriminator based on BERT were reported.

➔ **3rd run:**

We connected our two models - discriminator and generator into a one training loop (Inspired by GANs). However, we encountered some problems with RAM usage and loss propagation for gpt2.

➔ **Final run:**

We fixed all problems with the implementation and made more extensive data preprocessing. We added more detailed documentation and performed hyperparameter tuning. We pre-trained both models and then fine-tuned them in a GAN-like manner. The results are reported in the corresponding section.

Contributions of the teammates:

Adel has worked on:

- Data preprocessing for generator
- GPT2 generator pre-training
- Verifying the quality and realness of the generated tweets, i.e. checking that there are no copy-paste from the training set, so that the evaluator model is tested fairly
- Creating presentation of the project for final defense

Roman has worked on:

- Creating script to download and unpack data into according folders
- Data preprocessing for discriminator model
- Discriminator pre-training
- Training the evaluator model and measuring the results
- Finalizing the solution incorporating a github structure with a user-friendly notebooks running options

References

1. Kaggle competition
   https://www.kaggle.com/competitions/nlp-getting-started
2. GPT2 Text generation
   https://huggingface.co/gpt2?text=A+long+time+ago%2C
3. DistilBERT
   https://huggingface.co/docs/transformers/model_doc/distilbert
   +
   https://www.kaggle.com/code/alexia/kerasnlp-starter-notebook-disaster-tweets

4. Statistics of Fake Tweets
https://mediamakersmeet.com/19-42-of-active-twitter-accounts-are-fake-or-spam-analysis/