PMDL Project Deliverable 1.3

The third week of the project was dedicated to further enhancing the dataset of jokes and testing the initial generative model.

The model we decided to test first is GPT-2 recently published by the OpenAI community. The model is transformer-based and trained to predict the next word in a sequence on a large dataset. Since it is not designed specifically to generate jokes, a few of the last layers have to be retrained to suit our data. Despite successfully implementing the model on PyTorch we did not have sufficient resources and time to generate acceptable jokes for this week's report. However, we remain confident that after significantly more epochs the model will be able to produce human level content. So far, it has learned only basic word connections and even that is far from perfect.

The remaining two weeks we will focus on fine tuning the model. The screenshots of some parts of the code as well as the output are attached below.

```
REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-я #+_]')
STOPWORDS = set(stopwords.words('russian'))
# mystem = Mystem()

def preprocessing(text):
  # lowercase text
  text = text.lower()
  # replace REPLACE_BY_SPACE_RE symbols by space in text
  text = REPLACE_BY_SPACE_RE.sub(' ', text)
  # remove symbols which are in BAD_SYMBOLS_RE from text
  text = BAD_SYMBOLS_RE.sub('', text)
  # text = text.replace('x', '')

  # Single character removal
  text = re.sub(r"\s+[а-яА-Я]\s+", ' ', text)
  text = re.sub(r'<[^>]+>', '', text)

  # remove stopwors from text
  # tokens = mystem.lemmatize(text)
  tokens = text.split()
  tokens = [token for token in tokens if token not in STOPWORDS\
              and token != " " \
              and token.strip() not in punctuation]
  text = ' '.join(tokens)
  return text

df['text'] = df['text'].progress_apply(preprocessing)
```

```
100%|██████████| 138424/138424 [00:02<00:00, 48016.13it/s]
```

```
        outputs = model(work_jokes_tens, labels=work_jokes_tens)
        loss, logits = outputs[:2]
        loss.backward()
        sum_loss = sum_loss + loss.detach().data

        proc_seq_count = proc_seq_count + 1
        if proc_seq_count == BATCH_SIZE:
          proc_seq_count = 0
          batch_count += 1
          optimizer.step()
          scheduler.step()
          optimizer.zero_grad()
          model.zero_grad()

        if batch_count == 100:
          print(f"sum loss {sum_loss}")
          batch_count = 0
          sum_loss = 0.0

      # Store the model after each epoch to compare the performance of them
      torch.save(model.state_dict(), os.path.join(models_folder, f"gpt2_medium_joker_{epoch}.pt"))
```

```
EPOCH 0 started==============================
sum loss 3862.2001953125
EPOCH 1 started==============================
sum loss 3615.063232421875
EPOCH 2 started==============================
sum loss 3420.95703125
EPOCH 3 started==============================
sum loss 3290.638427734375
EPOCH 4 started==============================
sum loss 3195.344482421875
```