Computer Engineering Department

Natural Language Processing

Project Phase 2

Yasaman Lotfollahi
Ali Sedaghi

Table of contents

بخش اول

مشابه توضیحات درون لینک‌های قرار داده شده از سایت Hugging Face عمل می‌کنیم.

هایپر پارامترهای مدل تولید کننده جملات به صورت زیر می‌باشد:

```
MAX_LEN = 64
TRAIN_BATCH_SIZE = 64
VALID_BATCH_SIZE = 32
LEARNING_RATE = 1e-05
NUM_CLASSES = 6
```

ابتدا مدل از پیش آموخته BERT را بارگذاری می‌کنیم. سپس از BERT Tokenizer برای تبدیل متن String به Token استفاده می‌کنیم. یک تابع Detokenize برای عکس عمل یعنی Token به String نوشته شده است. از سه کاراکتر ویژه CLS و SEP و MASK نیز برای شروع، فاصله و نامعلومات استفاده کرده ایم.

```
model = BertForMaskedLM.from_pretrained('bert-base-uncased')

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', mask_toke="[MASK]", sep_token="[SEP]", pad_token="[PAD]")

def tokenize_batch(batch):
    return [tokenizer.convert_tokens_to_ids(sent) for sent in batch]

def untokenize_batch(batch):
    return [tokenizer.convert_ids_to_tokens(sent) for sent in batch]

def detokenize(sent):
    """ Roughly detokenizes (mainly undoes wordpiece) """
    new_sent = []
    for i, tok in enumerate(sent):
        if tok.startswith("##"):
            new_sent[len(new_sent) - 1] = new_sent[len(new_sent) - 1] + tok[2:]
        else:
            new_sent.append(tok)
    return new_sent

CLS = '[CLS]'
SEP = '[SEP]'
MASK = '[MASK]'
mask_id = tokenizer.convert_tokens_to_ids([MASK])[0]
sep_id = tokenizer.convert_tokens_to_ids([SEP])[0]
cls_id = tokenizer.convert_tokens_to_ids([CLS])[0]
```

از دیتاست پایتورچ برای تولید Batch های دیتاست استفاده شده است:

```
class Dataset(torch.utils.data.Dataset):
    def __init__(self, x):
        self.x = x

    def __getitem__(self, idx):
        return tokenizer(self.x[idx])["input_ids"]

    def __len__(self):
        return len(self.x)
```

لایه‌های اولیه مدل BERT که دارای فیچرهای مشترک و سطح پایین هستند را Freeze کرده ایم و فقط لایه‌های انتهایی آن را متناسب با هر کلاس دیتاست خود Fine Tune کرده ایم.

از روش Top K-Sampling که در درس NLG مورد بررسی قرار گرفت برای تولید رشته‌های هر کلاس استفاده کرده ایم.

```
def generate_step(out, gen_idx, top_k=0, sample=False, return_list=True):
    """ Generate a word from from out[gen_idx]

    args:
        - out (torch.Tensor): tensor of logits of size batch_size x seq_len x vocab_size
        - gen_idx (int): location for which to generate for
        - top_k (int): if >0, only sample from the top k most probable words
        - sample (Bool): if True, sample from full distribution. Overridden by top_k
    """
    # print("g", out["logits"].shape)
    logits = out["logits"][:, gen_idx]

    if top_k > 0:
        kth_vals, kth_idx = logits.topk(top_k, dim=-1)
        dist = torch.distributions.categorical.Categorical(logits=kth_vals)
        idx = kth_idx.gather(dim=1, index=dist.sample().unsqueeze(-1)).squeeze(-1)
    elif sample:
        dist = torch.distributions.categorical.Categorical(logits=logits)
        idx = dist.sample().squeeze(-1)
    else:
        idx = torch.argmax(logits, dim=-1)
    return idx.tolist() if return_list else idx
```

ماکزیمم طول کلمات درون یک جمله را برابر 15 قرار داده ایم. تابع Printer یک Wrapper بر روی Detokenize می‌باشد. درون تابع Generate روی Batch های دیتاست آموزش می‌دهیم و جملات تولید شده را با Ground Truth مقایسه می‌کنیم و از طریق تابع خطا Categorical Cross Entropy خطا را به شبکه بازگشت می‌دهیم تا مدل Fine Tune شود. ماکزیمم تکرار را برابر 500 قرار داده ایم.

```
def get_init_text(seed_text, max_len, batch_size = 1, rand_init=False):
    """ Get initial sentence by padding seed_text with either masks or random words to max_len """
    batch = [seed_text + [MASK] * max_len + [SEP] for _ in range(batch_size)]
    return tokenize_batch(batch)

def printer(sent, should_detokenize=True):
    if should_detokenize:
        sent = detokenize(sent)[1:-1]
    # print(" ".join(sent))
```

```python
def generate(n_samples, seed_text="[CLS]", batch_size=10, max_len=15, leed_out_len=15,
             sample=True, top_k=10, temperature=1.0, burnin=200, max_iter=500, print_every=1):
    sentences = []
    n_batches = math.ceil(n_samples / batch_size)
    start_time = time.time()
    seed_len = len(seed_text)
    batch = get_init_text(seed_text, max_len, batch_size)

    for ii in range(max_len):
        inp = [sent[:seed_len+ii+leed_out_len]+[sep_id] for sent in batch]
        inp = torch.tensor(batch).cuda()
#         torch.tensor(batch)
        out = model(inp)
        # print(seed_len, ii, out.keys())
        idxs = generate_step(out, gen_idx=seed_len+ii, top_k=top_k, sample=sample)
        for jj in range(batch_size):
            batch[jj][seed_len+ii] = idxs[jj]

    return untokenize_batch(batch)
```

در ادامه نتیجه مدل برای تولید جملات برای شخصیت Rachel آورده شده است:

```
[CLS] wait wait wait wait think what ##s going make take real time person say good night ross said totally yeah know well know know real time maybe watching movies see know woman ##s interest joey know pretty much anything [SEP]
[CLS] think possible can ##t well go ##n see right away joey come back walks behind joey goes outside door joey goes outside door joey says take walk out door looking joey go ##n worry joey joey try alright alright [SEP]
[CLS] maybe isn ##t time go ##n get rid time right can ##t wait hours remember need time watch new show long time ago show little show coming show play time emma getti ady make emma ready play show play time [SEP]
[CLS] know that ##s right little girl know wow know baby made joey ##s wish never sex ever made love ross got lost anything joey first ever knew guys way way sure coul joey first want well yeah know know [SEP]
[CLS] good luck bed going ross made promise tomorrow tomorrow gavin left early presentation time went bed going walking outside window building goes bed going building ow building side let break legs look around bedroom door trying time light up window [SEP]
```

+ Code + Markdown

بخش دوم

هایپر پارامترهای مدل به صورت زیر می‌باشد:

```
MAX_LEN = 16
TRAIN_BATCH_SIZE = 64
VALID_BATCH_SIZE = 32
LEARNING_RATE = 1e-3
NUM_CLASSES = 6
EPOCHS = 500
```

همچنین دیتاست را به نسبت 80 درصد تقسیم می‌کنیم:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

```
print(f"x_train: {x_train.shape}")
print(f"y_train: {y_train.shape}")
print(f"x_test: {x_test.shape}")
print(f"y_test: {y_test.shape}")
```

```
x_train: (7104,)
y_train: (7104,)
x_test: (1776,)
y_test: (1776,)
```

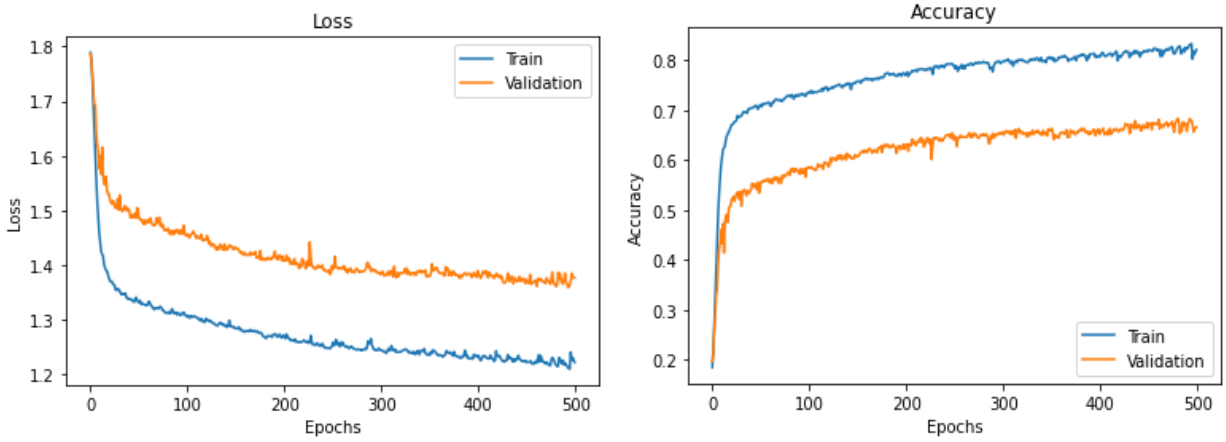
از GloVe Vectorizer جهت تبدیل کلمات به بردارهای اعداد استفاده می‌کنیم.

با استفاده از Datasets درون PyTorch دیتاست خود را ایجاد می‌کنیم.

با استفاده از پایتورچ یک مدل ساده که معماری آن به صورت زیر است ایجاد می‌کنیم. این مدل از یک لایه امبدینگ با سایز 300، یک لایه LSTM با اندازه 32، یک لایه Linear به اندازه 64، یک لایه Batch Norm، یک تابع فعالسازی Tanh، یک لایه Linear دیگر با اندازه 64 و یک لایه فعالسازی Softmax با اندازه 6 (تعداد کلاس) تشکیل شده است. از تابع خطا Cross Entropy و بهینه‌ساز Adam استفاده شده است.

```python
class ClassificationModel(nn.Module):
    def __init__(self, num_class=6, hidden_dim=32, embed_size=300, lstm_layers=1, input_size=64, fc_size=64):
        super(ClassificationModel, self).__init__()
        self.embed_size = embed_size
        self.lstm_layers = lstm_layers
        self.hidden_dim = hidden_dim

        self.lstm = nn.LSTM(input_size=self.embed_size, hidden_size=self.hidden_dim, num_layers=self.lstm_layers, batch_first=True)

        self.linears = nn.Sequential(
            nn.Linear(self.hidden_dim, 64),
            nn.BatchNorm1d(64),
            nn.Tanh(),
            nn.Linear(64, num_class),
        )



    def forward(self, x):
        out, (hidden, cell) = self.lstm(x)
        out = out[:, -1, :] # using the last
        out = self.linears(out)

        return F.softmax(out, dim=1)

model = ClassificationModel()
```

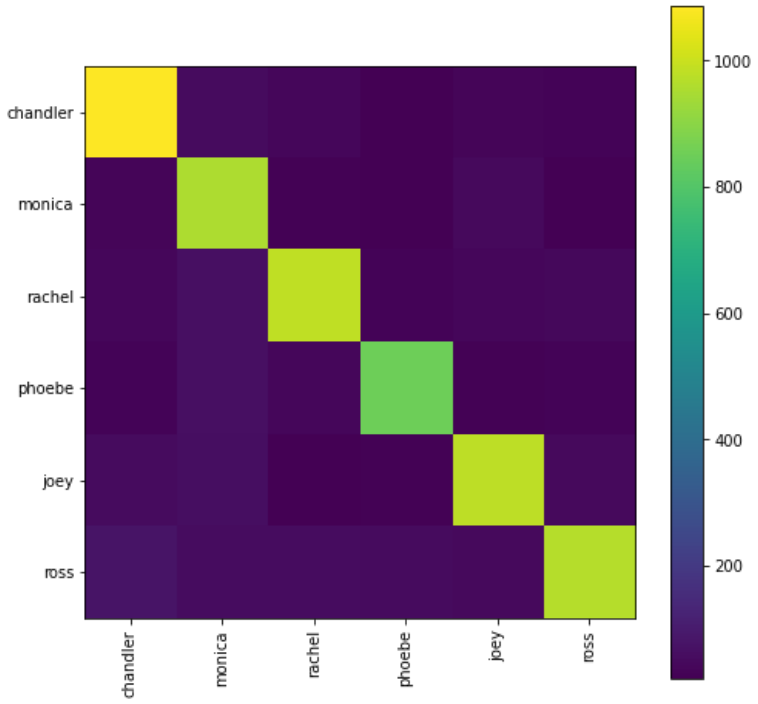
مدل را در 500 ایپاک آموزش می‌دهیم که نتایج آن به صورت زیر است:




```
| end of epoch 495 | train loss 1.210 | val loss 1.365 | train acc 0.833 | val acc 0.679
-----------------------------------------------------------------------------------------
| end of epoch 496 | train loss 1.241 | val loss 1.371 | train acc 0.803 | val acc 0.671
-----------------------------------------------------------------------------------------
| end of epoch 497 | train loss 1.226 | val loss 1.385 | train acc 0.818 | val acc 0.657
-----------------------------------------------------------------------------------------
| end of epoch 498 | train loss 1.231 | val loss 1.378 | train acc 0.812 | val acc 0.664
-----------------------------------------------------------------------------------------
| end of epoch 499 | train loss 1.226 | val loss 1.378 | train acc 0.817 | val acc 0.664
-----------------------------------------------------------------------------------------
| end of epoch 500 | train loss 1.223 | val loss 1.377 | train acc 0.821 | val acc 0.666
-----------------------------------------------------------------------------------------
```

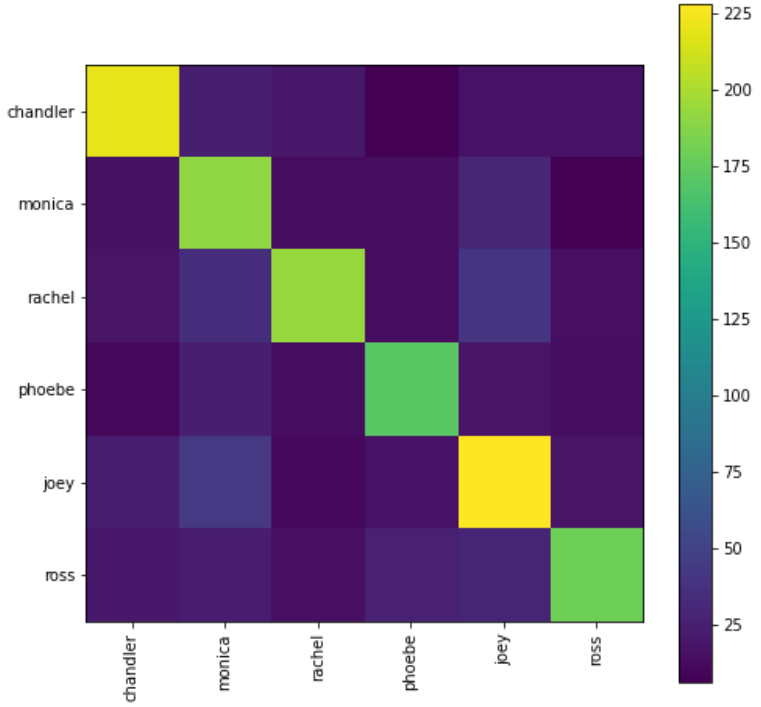
متریک‌های F1، Recall، Precision و Support و همچنین Accuracy و Confusion Matrix در فاز Train:

```
Classification Report Train
              precision    recall  f1-score   support

    chandler       0.82      0.85      0.84      1273
      monica       0.76      0.86      0.81      1113
      rachel       0.84      0.82      0.83      1207
      phoebe       0.84      0.81      0.83      1050
        joey       0.83      0.82      0.82      1196
        ross       0.84      0.77      0.80      1265

    accuracy                           0.82      7104
   macro avg       0.82      0.82      0.82      7104
weighted avg       0.82      0.82      0.82      7104

Confusion Matrix Train
[[1087   54   39   25   36   32]
 [  36  957   29   22   47   22]
 [  41   64  986   32   40   44]
 [  33   64   39  852   29   33]
 [  53   63   23   28  981   48]
 [  77   56   59   52   51  970]]
```



متریک‌های F1، Recall، Precision و Support و همچنین Accuracy و Confusion Matrix در فاز Validation:

```
Classification Report Validation
              precision    recall  f1-score   support

    chandler       0.71      0.72      0.71       305
      monica       0.56      0.71      0.63       269
      rachel       0.73      0.62      0.67       315
      phoebe       0.70      0.68      0.69       252
        joey       0.63      0.67      0.65       342
        ross       0.72      0.61      0.66       293

    accuracy                           0.67      1776
   macro avg       0.67      0.67      0.67      1776
weighted avg       0.67      0.67      0.67      1776

Confusion Matrix Validation
[[220  25  20   6  17  17]
 [ 16 191  13  13  30   6]
 [ 18  35 194  13  40  15]
 [ 11  25  13 171  18  14]
 [ 25  43  11  17 228  18]
 [ 21  23  15  26  29 179]]
```



بخش سوم

برای این قسمت از مدل‌های مبتنی بر Transformer درون Hugging Face استفاده شده است.

هایپر پارامترهای مدل به صورت زیر می‌باشد:

```
MAX_LEN = 64
TRAIN_BATCH_SIZE = 64
VALID_BATCH_SIZE = 32
LEARNING_RATE = 1e-05
NUM_CLASSES = 6
EPOCHS = 10
```

مشابه قسمت دوم دیتاست را آماده می‌کنیم.

این بار به جای GloVe از Distil BERT Tokenizer استفاده کردیم که قوی‌تر و پیشرفته‌تر می‌باشد و دارای Attention Mask نیز می‌باشد.

DistilBERT Tokenizer

```
[9]  tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-uncased")

x_train_tokenized = tokenizer(x_train.tolist(), padding=True, truncation=True, max_length=MAX_LEN)
x_test_tokenized = tokenizer(x_test.tolist(), padding=True, truncation=True, max_length=MAX_LEN)
```

```
Downloading: 100% 226k/226k [00:00<00:00, 279kB/s]
Downloading: 100% 28.0/28.0 [00:00<00:00, 810B/s]
Downloading: 100% 483/483 [00:00<00:00, 13.3kB/s]
```

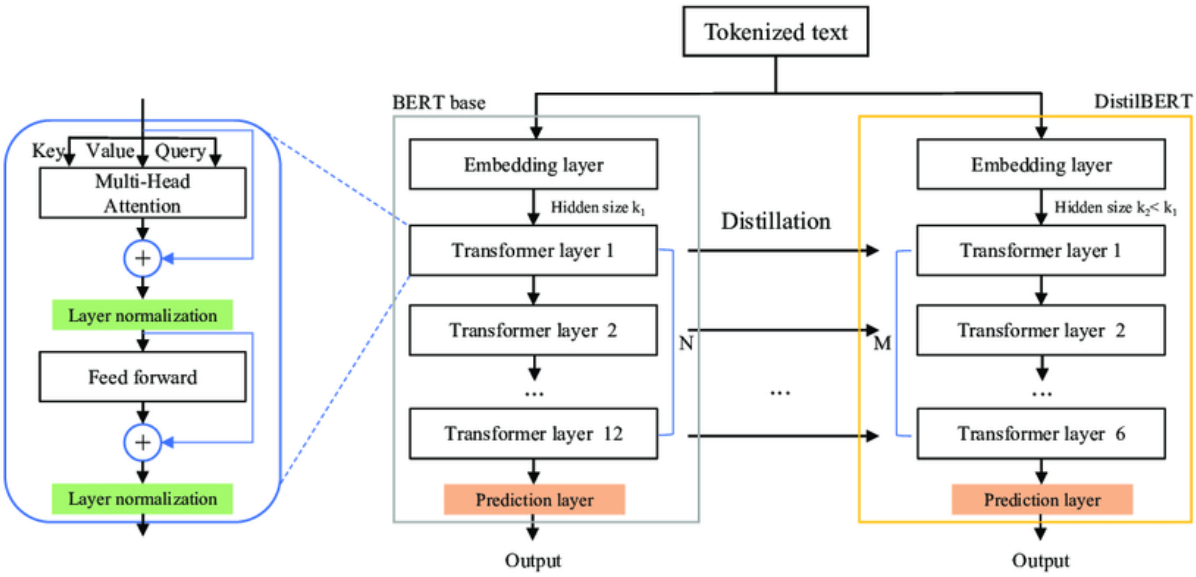
همچنین از مدل Distil BERT Seq Classifier که از پیش آموزش دیده شده است نیز استفاده کردیم که مبتنی بر ساختار Transformer است.

Model

```
[11]  model = DistilBertForSequenceClassification.from_pretrained(
          "distilbert-base-uncased",
          num_labels=NUM_CLASSES
      )
```

```
Downloading: 100% 256M/256M [00:04<00:00, 61.2MB/s]
```

معماری این مدل به صورت زیر است که همانطور که مشاهده می‌شود از ساختار Transformer و همچنین Multi Head Attention بهره می‌برد.

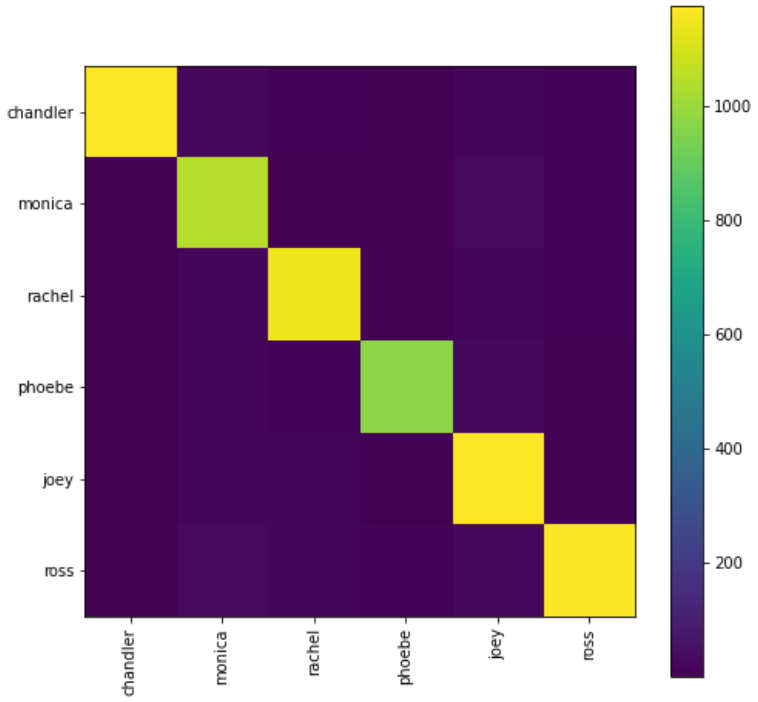


مدل را در 10 ایپاک آموزش می‌دهیم و نتایج و متریک‌های آن برای هر ایپاک به صورت زیر است:

Epoch	Training Loss	Validation Loss	Accuracy	Precision	F1
1	1.763900	1.692776	0.295608	0.362175	0.268334
2	1.521100	1.407373	0.469032	0.478789	0.461560
3	1.049900	1.080330	0.626689	0.634673	0.623860
4	0.642700	0.866115	0.717342	0.727719	0.718102
5	0.407900	0.775580	0.770270	0.779488	0.770904
6	0.298500	0.744298	0.786599	0.795118	0.787742
7	0.239000	0.735202	0.796171	0.798505	0.796475
8	0.207800	0.743849	0.801239	0.803646	0.801275
9	0.187500	0.748156	0.806306	0.809379	0.806430
10	0.172500	0.733611	0.807995	0.809453	0.807956

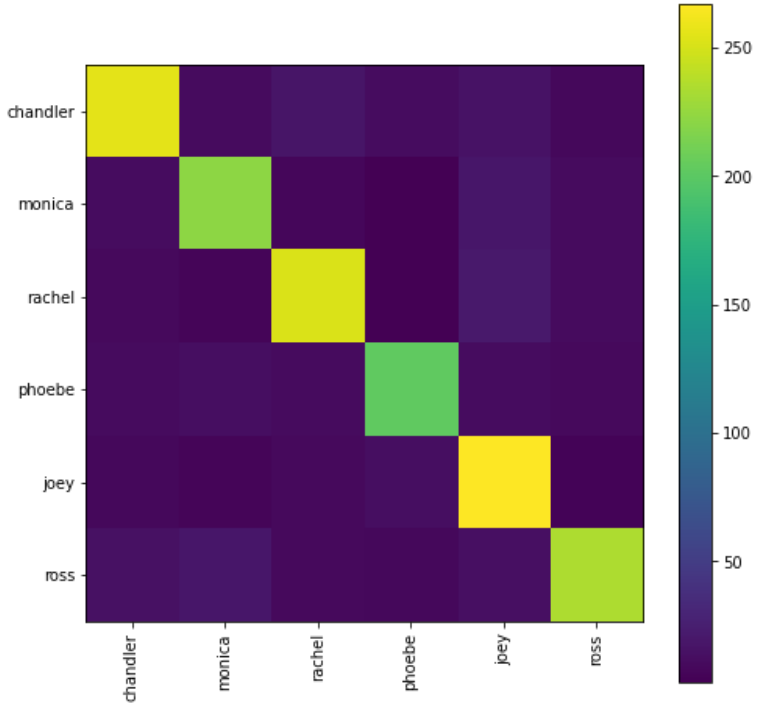
متریک‌های F1، Recall، Precision و Support و همچنین Accuracy و Confusion Matrix در فاز Train:

```
Classification Report Train
              precision    recall  f1-score   support

    chandler       0.99      0.94      0.96      1256
      monica       0.90      0.94      0.92      1107
      rachel       0.94      0.94      0.94      1217
      phoebe       0.97      0.93      0.95      1042
        joey       0.90      0.96      0.93      1226
        ross       0.96      0.93      0.95      1256

    accuracy                           0.94      7104
   macro avg       0.94      0.94      0.94      7104
weighted avg       0.94      0.94      0.94      7104

Confusion Matrix Train
[[1176   25   14    6   22   13]
 [   4 1046   10    3   33   11]
 [   4   21 1148   10   23   11]
 [   3   23   11  971   28    6]
 [   1   19   20    5 1173    8]
 [   5   31   16   11   24 1169]]
```



متریک‌های F1، Recall، Precision و Support و همچنین Accuracy و Confusion Matrix در فاز Validation:

```
Classification Report Validation
              precision    recall  f1-score   support

    chandler       0.82      0.80      0.81       322
      monica       0.80      0.81      0.80       275
      rachel       0.82      0.83      0.82       305
      phoebe       0.83      0.78      0.81       260
        joey       0.77      0.86      0.81       312
        ross       0.83      0.78      0.80       302

    accuracy                           0.81      1776
   macro avg       0.81      0.81      0.81      1776
weighted avg       0.81      0.81      0.81      1776

Confusion Matrix Validation
[[256  11  18  12  16   9]
 [ 12 222   8   3  19  11]
 [ 10   7 252   4  21  11]
 [ 11  13  11 203  12  10]
 [  9   7  10  13 267   6]
 [ 15  19  10   9  14 235]]
```



مقایسه بخش دوم و سوم

معیارهای زیر برای دو مدل بخش دوم و سوم و به صورت Weighted محاسبه شده است تا Unbalance بودن کلاس‌های دیتاست تاثیرگذار نباشد.

	Accuracy	Precision	Recall	F1
Simple Model	Train: 0.82 Val: 0.67	Train: 0.82 Val: 0.67	Train: 0.82 Val: 0.67	Train: 0.82 Val: 0.67
Transformer	Train: 0.82 Val: 0.67	Train: 0.82 Val: 0.67	Train: 0.82 Val: 0.67	Train: 0.82 Val: 0.67

همانطور که مشاهده می‌شود مدل پیشرفته و حاوی Transformer در تمامی معیارها عملکرد مطلوب‌تری داشته است.

پیش‌بینی‌های مدل

bill colleen
P: chandler
T: chandler
---
mhmm
P: rachel
T: monica
---
sure
P: rachel
T: rachel

Please these guys we haven't even moved yet picking china patterns mike seems gag little and laughs nervously begin leave phoebe bolts back
P: phoebe
T: phoebe
---
don't sing sit pass judgments others
P: monica
T: monica
---

whats matter
P: joey
T: rachel
---
look ross he's sandy sensitive thats
P: rachel
T: rachel
---
mikes
P: phoebe
T: phoebe
---
know mine
P: monica
T: monica
---
know don't mind male nanny draw line male wetnurse laugh even fake
P: chandler
T: chandler
---
weird think he's gross yet you're willing eat crackers mike desperately tries get rid crackers mouth uses hand clean tongue
P: phoebe
T: phoebe
---
going next argument
P: phoebe
T: ross
---
much thinking
P: monica
T: monica
---
good news quickly wringing hands course good news said deadpan doctor connelly called good news would said excitedly doctor connelly called problem problem
P: monica
T: monica
---
yeah
P: joey
T: rachel
---

hes smart he's qualified give one good reason shouldn't try
P: rachel
T: rachel
---
cause good money doesn't change fact evil blood sucking corporate machine
P: monica
T: phoebe
---
really sure
P: chandler
T: chandler
---
come ben
P: rachel
T: monica
---
gon stop right glenda okay look like first time huh want twos want back
P: rachel
T: ross
---
yeah want tickets takes bowl rachael buying knicks steffi graf
P: joey
T: joey
---
yeah know need hugsy don't worry emma totally understand wont whatever leaves room
P: rachel
T: rachel
---
coming bedroom say goodbye elves tulsa
P: chandler
T: chandler
---
ooh yeah going anywhere
P: joey
T: joey
---
monicathink washed hands
P: ross
T: ross
---
thanks
P: chandler
T: joey
---

still
P: ross
T: ross
---
woman can't know work shes friend mine made big stink awful massage chains
P: phoebe
T: phoebe
---
well depends
P: chandler
T: phoebe
---
still things
P: monica
T: monica
---
going next argument
P: phoebe
T: ross
---
original
P: rachel
T: rachel

منابع

python - Why sklearn returns the accuracy and weighted-average recall the same value in binary classification? - Stack Overflow

https://huggingface.co/HooshvareLab/bert-fa-base-uncased

https://huggingface.co/docs/transformers/training