

پروژه درس هوش محاسباتی

موضوع:

Social Media Sentiment

سبا عامری-معصومه مختاری

Dataset:

دیتاست استفاده شده در این پروژه از توییت هایی استخراج شده است که در آن ها مردم نظرات خود را در مورد چند شرکت هواپیمایی بیان کرده اند. تمام داده ها به سه دسته تقسیم شده اند:

1. positive

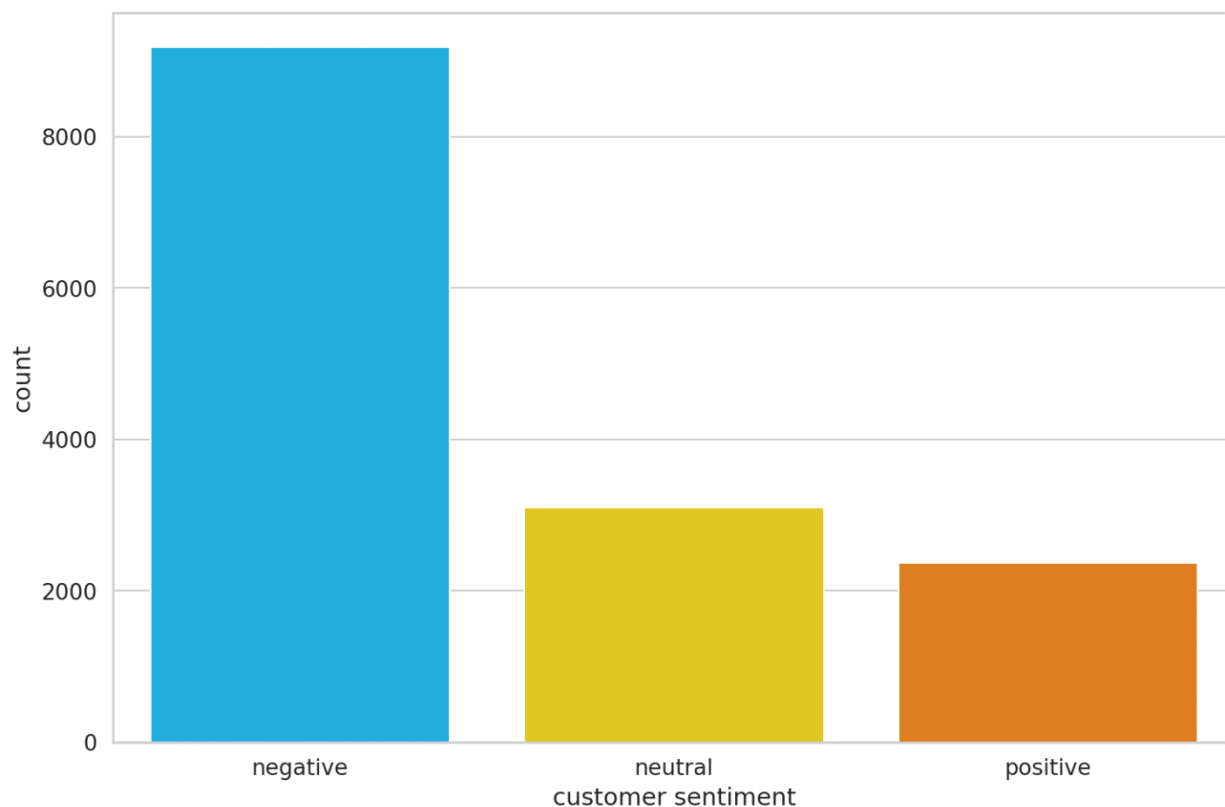
2. neutral

3. negative

تمام داده ها در یک فایل csv ذخیره شده اند و هر سلول دارای اطلاعاتی مانند متن توییت، نام شخص توییت کننده، sentiment توییت، درصد sentiment confidence و غیره می باشد. در تصویر زیر برخی از این پارامترها را مشاهده می کنید:

```
[ ] df = pd.read_csv('Airline-Sentiment-2-w-AA.csv', encoding= 'unicode_escape')
df.head(10)
```

	_unit_id	_golden	_unit_state	_trusted_judgments	_last_judgment_at	airline_sentiment	airline_sentiment:confidence	negativereason	negativereason:confidence	airline	airline_s
0	681448150	False	finalized	3	2/25/15 5:24	neutral	1.0000	NaN	NaN	Virgin America	
1	681448153	False	finalized	3	2/25/15 1:53	positive	0.3486	NaN	0.0000	Virgin America	
2	681448156	False	finalized	3	2/25/15 10:01	neutral	0.6837	NaN	NaN	Virgin America	
3	681448158	False	finalized	3	2/25/15 3:05	negative	1.0000	Bad Flight	0.7033	Virgin America	
4	681448159	False	finalized	3	2/25/15 5:50	negative	1.0000	Can't Tell	1.0000	Virgin America	
5	681448162	False	finalized	3	2/25/15 9:10	negative	1.0000	Can't Tell	0.6842	Virgin America	



همانطور که مشاهده می کنید مقدار دیتا های negative بسیار بیشتر از دو گروه دیگر است و دلیل آن این است که بیشتر کاربران مشکلات شرکت های هواپیمایی را بیان کرده اند و در نتیجه نظر منفی بیشتری داده اند.

BERT:

ما برای تحلیل و آموزش (train) دادن داده های خود از کتابخانه **BERT from the Hugging Face transformer library** استفاده می کنیم.

BERT مخفف *Bidirectional Encoder Representations from Transformers* می باشد و با توجه به اسم آن یکی از ویژگی های مهم آن همان تحلیل دو طرفه داده های متنی می باشد.

مدل BERT توسط masked language modeling (MLM) و next sentence prediction (NSP) تمرین داده شده است.

مدل BERT یکی از بهترین ابزارها در زمینه تحلیل Sentimental Analysis است. با استفاده از مدل های pre-trained این کتابخانه می توان بسیار از مراحل train داده را با سرعت بیشتری انجام داد. ما در این پروژه از مدل کوچک تر آن استفاده می کنیم تا داده های تمرین داده مان سریع تر آماده شوند و به نسبت حجم کمتری داشته باشند.

علاوه بر آن از کتابخانه PyTorch برای تحلیل داده ها استفاده می کنیم.

تابع های نوشته شده برای Text Preprocessing:

```
[ ] class GPReviewDataset(data.Dataset):

    def __init__(self, reviews, targets, tokenizer, max_len):
        self.reviews = reviews
        self.targets = targets
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.reviews)

    def __getitem__(self, item):
        review = str(self.reviews[item])
        target = self.targets[item]

        encoding = self.tokenizer.encode_plus(
            review,
            max_length = self.max_len,
            add_special_tokens = True,
            pad_to_max_length = True,
            return_attention_mask = True,
            return_token_type_ids = False,
            return_tensors = 'pt'
        )

        return {
            'review_text': review,
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'targets' : torch.tensor(target, dtype=torch.long)
        }
```

```
def create_data_loader(df, tokenizer, max_len, batch_size):
    ds = GPReviewDataset(
        reviews = df.text.to_numpy(),
        targets=df.sentiment.to_numpy(),
        tokenizer = tokenizer,
        max_len = max_len
    )

    return data.DataLoader(
        ds,
        batch_size=batch_size,
        num_workers=5
    )
```

Training

در این مرحله قبل از شروع train داده ها ابتدا آن ها را به سه گروه برای تمرین دادن، ارزیابی و تست تقسیم بندی می کنیم. تعداد داده های موجود در گروه train حدود 11 هزار تا است و به میزان قابل توجهی از دو گروه دیگر بیشتر است. دو گروه دیگر هر کدام به انداز 1400 نمونه در خود دارند.

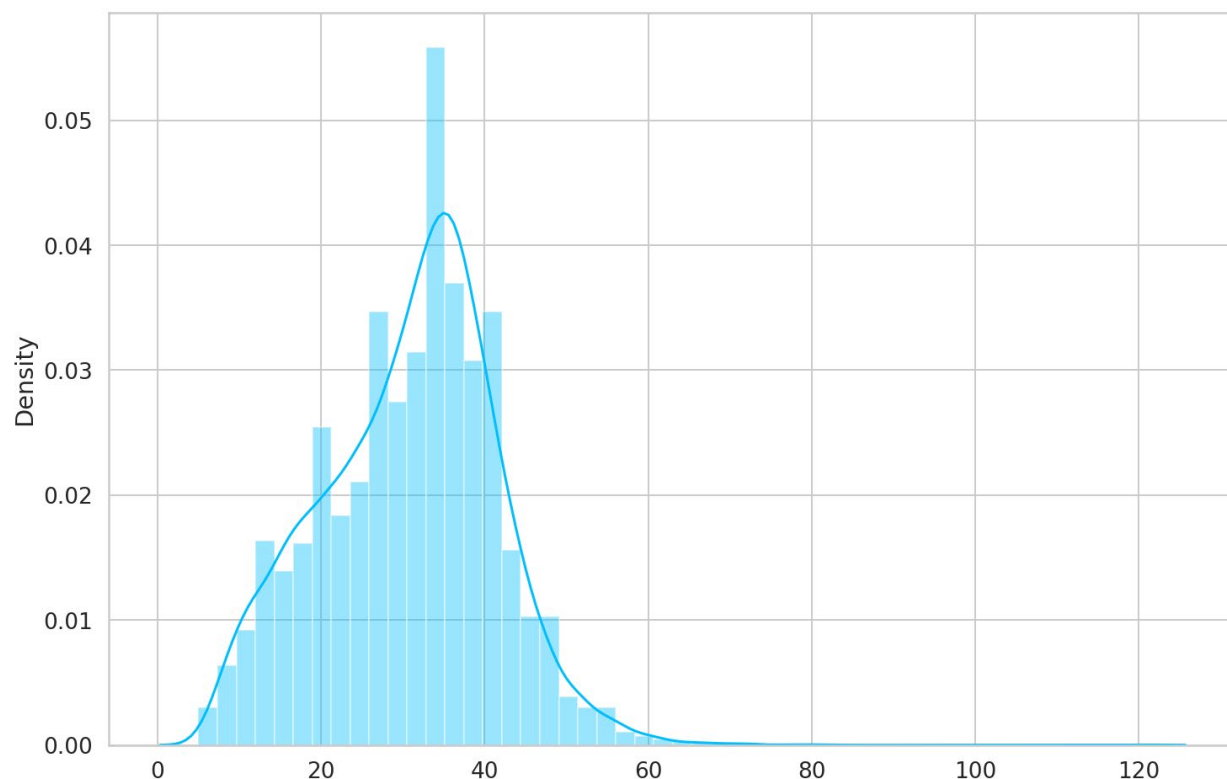
```
[ ] df_train, df_test = train_test_split(df, test_size=0.2, random_state=RANDOM_SEED)
    df_val, df_test = train_test_split(df_test, test_size=0.5, random_state=RANDOM_SEED)

[ ] df_train.shape, df_val.shape, df_test.shape

((11712, 21), (1464, 21), (1464, 21))
```

```
[ ] train_data_loader = create_data_loader(df_train, tokenizer, MAX_LEN, BATCH_SIZE)
    val_data_loader = create_data_loader(df_val, tokenizer, MAX_LEN, BATCH_SIZE)
    test_data_loader = create_data_loader(df_test, tokenizer, MAX_LEN, BATCH_SIZE)
```

طول بیشینه ای که برای داده های خود انتخاب کردیم 60 (بر اساس نمودار زیر)، طول batch 16 و تعداد epoch ها 12 تا است.



```
[ ] model(input_ids, attention_mask)
```

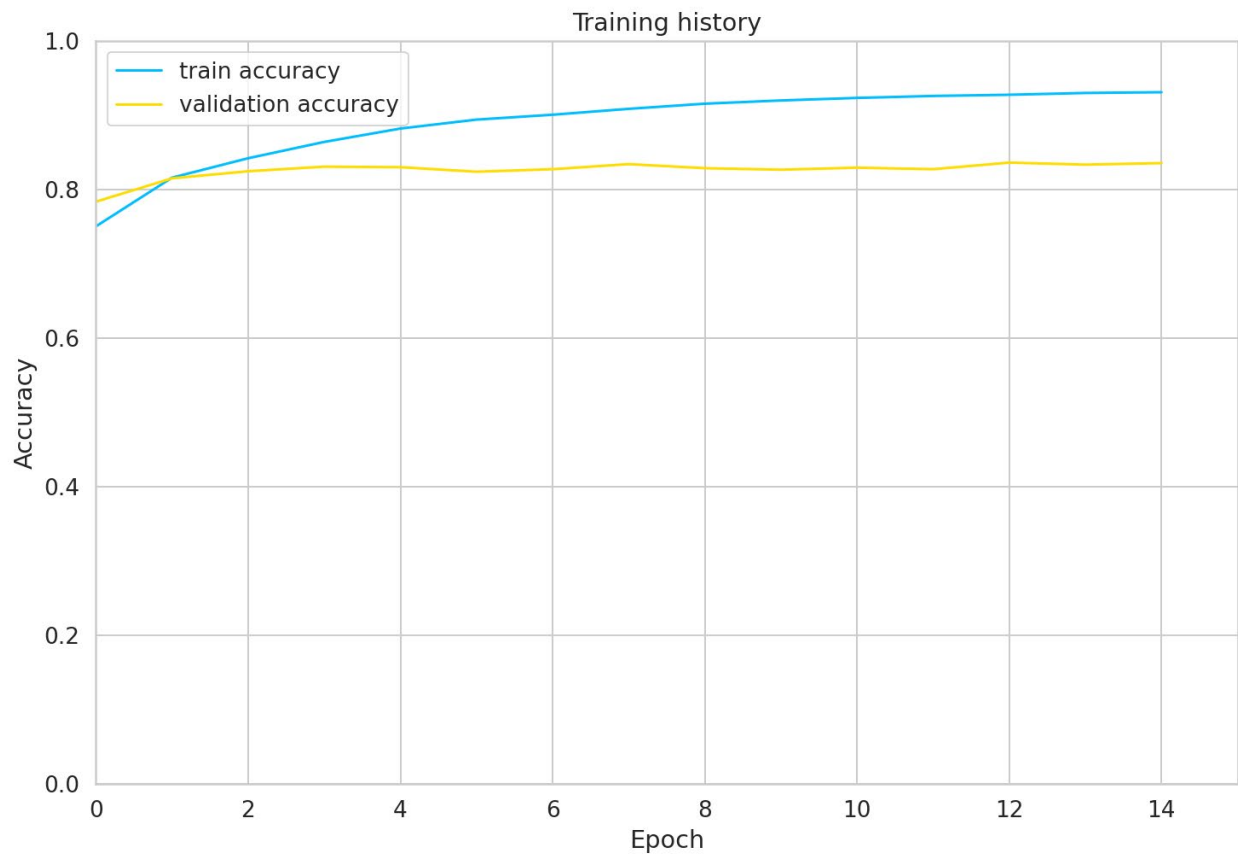
```
tensor([[0.2183, 0.4848, 0.2969],  
        [0.2492, 0.3111, 0.4398],  
        [0.3174, 0.2836, 0.3990],  
        [0.2061, 0.4039, 0.3900],  
        [0.5320, 0.2592, 0.2088],  
        [0.2275, 0.4242, 0.3484],  
        [0.2756, 0.3216, 0.4028],  
        [0.3619, 0.2796, 0.3585],  
        [0.4033, 0.2013, 0.3954],  
        [0.3190, 0.2150, 0.4660],  
        [0.3291, 0.2339, 0.4371],  
        [0.3617, 0.1929, 0.4454],  
        [0.2381, 0.3660, 0.3960],  
        [0.2568, 0.3170, 0.4261],  
        [0.3067, 0.2758, 0.4176],  
        [0.1525, 0.3738, 0.4737]], device='cuda:0', grad_fn=<SoftmaxBackward>)
```

همانطور که در تصویر بالا مشاهده می کنید 16 نمونه داریم و در هر کدام مشخص شده است که مدل ما با چه درصد احتمالی برای هر کلاس بررسی انجام می دهد.

بعد از آن داده های خود را train می دهیم، که به مدت epoch 15 انجام می شود.

Evaluation:

حال بعد از train داده ها، باید ارزیابی آن ها انجام پذیرد. نتیجه ارزیابی به صورت زیر خواهد بود:



```
[ ] print(classification_report(y_test, y_pred, target_names=class_names))
```

	precision	recall	f1-score	support
negative	0.89	0.92	0.90	959
neutral	0.75	0.65	0.70	293
positive	0.76	0.79	0.77	212
accuracy			0.85	1464
macro avg	0.80	0.79	0.79	1464
weighted avg	0.84	0.85	0.84	1464

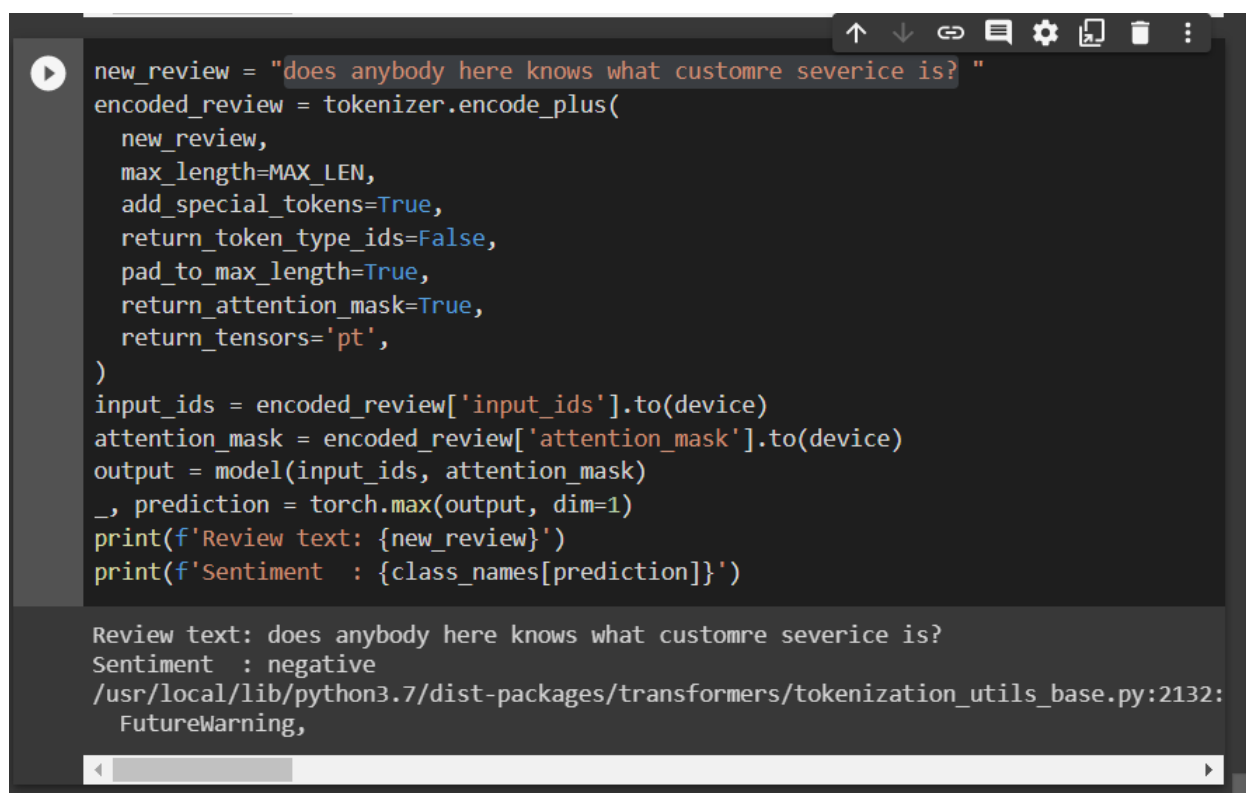
با توجه به تصویر بالا که میزان دقت مدل را بعد از epoch 15 نشان می دهد، می توان مشاهده که دقت مدل در تشخیص sentiment منفی بیشتر است چون مقدار داده های موجود برای این گروه بیشتر است.

Testing:

در مرحله ی آخر با مثال هایی توانایی مدل خود را مورد بررسی قرار می دهیم. جمله ی نمونه ی ما به صورت زیر است:

مثال اول:

does anybody here knows what customre severice is?



```
new_review = "does anybody here knows what customre severice is? "
encoded_review = tokenizer.encode_plus(
    new_review,
    max_length=MAX_LEN,
    add_special_tokens=True,
    return_token_type_ids=False,
    pad_to_max_length=True,
    return_attention_mask=True,
    return_tensors='pt',
)
input_ids = encoded_review['input_ids'].to(device)
attention_mask = encoded_review['attention_mask'].to(device)
output = model(input_ids, attention_mask)
_, prediction = torch.max(output, dim=1)
print(f'Review text: {new_review}')
print(f'Sentiment : {class_names[prediction]}')
```

Review text: does anybody here knows what customre severice is?
Sentiment : negative
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2132: FutureWarning,

مثال دوم:

"loved the flights staff"


```
[ ] new_review = "loved the flights staff"
    encoded_review = tokenizer.encode_plus(
        new_review,
        max_length=MAX_LEN,
        add_special_tokens=True,
        return_token_type_ids=False,
        pad_to_max_length=True,
        return_attention_mask=True,
        return_tensors='pt',
    )
    input_ids = encoded_review['input_ids'].to(device)
    attention_mask = encoded_review['attention_mask'].to(device)
    output = model(input_ids, attention_mask)
    _, prediction = torch.max(output, dim=1)
    print(f'Review text: {new_review}')
    print(f'Sentiment : {class_names[prediction]}')
```

```
Review text: loved the flights staff
Sentiment : positive
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2132:
FutureWarning,
```

همانطور که مشاهده می کنید، مدل sentiment جمله را به درستی تشخیصی داده است.

لینک ویدیو توضیحات پروژه:

https://drive.google.com/file/d/1vQbxae5eAZ5Uu_uwXAucxwnei3p7n1tZ/view?usp=sharing