



درس مباني يادگيري عميق
گزارش پروژه پاياني

موضوع: تحليل احساسات در متن فارسي

استاد درس : دکتر مرضيه داوودآبادی

تهيه کنندگان: علی سلطانی و ريحانه هاشم زاده کلهرودی

دانشگاه علم و صنعت ايران، دانشکده مهندسي کامپيوتر

نيمسال اول تحصيلي ۱۴۰۲ - ۱

موضوع:

در این پروژه قصد داریم با تحلیل متن که به زبان فارسی هست احساسات افراد را تشخیص داده و دسته بندی کنیم.

مقدمه:

تحلیل احساسات یکی از زیرشاخه‌های پردازش زبان طبیعی (NLP) است که با استفاده از هوش مصنوعی و یادگیری ماشین، قادر است احساسات و نگرش‌های موجود در متون را شناسایی و دسته‌بندی کند. این فناوری می‌تواند در بسیاری از زمینه‌ها مانند تحلیل بازخورد مشتریان، نظارت بر شهرت آنلاین، تشخیص خبرهای جعلی و غیره کاربرد داشته باشد.

برای تحلیل احساسات در متون فارسی، چالش‌های خاصی وجود دارد که باید در نظر گرفته شوند. برخی از این چالش‌ها عبارتند از:

- کمبود داده‌های برچسب‌گذاری شده به زبان فارسی برای آموزش مدل‌های یادگیری ماشین
- وجود اصطلاحات، ضرب‌المثل‌ها، کنایه‌ها و زبان عامیانه که ممکن است معنای متن را تغییر دهند
- وجود اختلافات لهجه‌ای، املایی و نگارشی در متون فارسی
- وجود حروف و کلمات عربی، انگلیسی و دیگر زبان‌ها در متون فارسی
- وجود نیاز به تشخیص و تجزیه و تحلیل احساسات خاص مانند شادی، غم، عصبانیت و غیره

برای رفع این چالش‌ها، روش‌ها و رویکردهای مختلفی ارائه شده‌اند که می‌توانند از روش‌های مبتنی بر قانون، یادگیری ماشین، یادگیری عمیق، روش‌های ترکیبی و غیره باشند. هر روش مزایا و معایب خود را دارد و بسته به نوع و حجم داده‌ها، هدف و دقت مورد نظر، می‌توان از آن‌ها استفاده کرد.

در این گزارش، ما قصد داریم با بررسی و حل چالش‌ها با روش‌های یادگیری عمیق را برای تحلیل احساسات در متون فارسی معرفی و ارزیابی کنیم

مجموعه داده:

برای این که بتوانیم مدل را آموزش بدهیم نیاز به داده مناسب داریم ازاونجایی که ورودی جملات و خروجی آن نوع احساسات هست که شامل هفت دسته نفرت تعجب ترس خشم ناراحت خوشحال و سایر هست، به این منظور از دیتا ست armanemo استفاده کردیم که دارای جملات با لیبل‌های احساسات بود*.

برای اینکه بتوان داده ها را استفاده کرده با استفاده از مقاله ای که خودشون در اختیار قرار دادند، به پیش پردازش و ... پرداختیم که در ادامه آن را مشاهده میکنیم.

پیش پردازش داده ها:

پیش پردازش متن فارسی یکی از مراحل مهم در پردازش زبان طبیعی فارسی است که شامل تبدیل، تصحیح، استخراج و ساختاردهی اطلاعات موجود در متن است. به ترتیب مراحل زیر باید برای آن اجرا شود:

نرمال سازی متن: این روش شامل جایگزین کردن حروف مختلف با یک حرف استاندارد، حذف فاصله‌های اضافی، اصلاح اشتباهات املایی و تبدیل اعداد به کلمات است.

توکن سازی متن: این روش شامل شکستن متن به واحدهای کوچکتر مانند کلمات، جملات، پاراگراف‌ها و غیره است.

ریشه یابی و لم سازی متن: این روش شامل برداشتن پسوندها و پیشوندها از کلمات و بازگرداندن آن‌ها به ریشه یا لم اصلی‌شان است.

برچسب زنی نقش کلمات و چانک سازی متن: این روش شامل تعیین نقش دستوری و عملکرد کلمات در جمله و گروه بندی کلمات مرتبط با هم است.

تجزیه و تحلیل وابستگی متن: این روش شامل شناسایی روابط بین کلمات در جمله و ساختار درختی جمله است.

اصلاح نگارش متن: این روش شامل بهبود قواعد نگارشی، املایی، نحوی و معنایی متن است.

برای انجام این کار، می‌توان از کتابخانه‌های مختلف پایتون استفاده کرد.

بعد از بررسی کتابخانه های farasa, stanza , parsivar, persiontool, hazam کتابخانه یارسی ور را انتخاب کردیم.

پارسیور: یکی از کتابخانه‌های معروف و کارآمد برای پیش پردازش متن فارسی، کتابخانه پارسیور (Parsivar) است که شامل ابزارهای مختلفی مانند نرمال‌ساز، توکن‌ساز، ریشه‌یاب، لم‌ساز، برچسب‌زن، چانک‌ساز، تجزیه‌گر و املایی است.

به صورت دقیق تر مراحل اجرا شده شامل این مراحل هست:

1. هر کاراکتر انگلیسی موجود را حذف کردیم
2. کلماتی با تکرار حروف برای تاکید بیشتر را حذف کردیم
3. هر حروف عربی و مرتبط به آن را حذف کردیم
4. هر کاراکتر و کلمه ای که به صورت کلی فارسی نباشد
5. هشتگ ها را حذف کردیم منتها محتوای هشتگ را نگه داشتیم
6. شماره های فارسی هم را حذف کردیم.

که به ترتیب در زیر آمده اند:

```
# Initialize Parsivar normalizer
normalizer = Normalizer()

# Function to perform additional pre-processing steps
def additional_preprocessing(text):
    # Remove English characters
    text = re.sub(r'[a-zA-Z]', '', text)

    # Remove repeated letters more than twice in non-standard Persian words
    text = re.sub(r'(\.|\{2,}|,| r'\1\1', text)

    # Remove Arabic diacritics
    text = re.sub(r'[\u064B-\u065F\u0670]', '', text)

    # Remove remaining non-Persian characters
    text = re.sub(r'^[٩-ٲ-ٳ-ٴ-ٵ]', '', text)

    # Remove hashtag sign while preserving hashtag information
    text = re.sub(r'#(\w+)', r'\1', text)

    # Remove Persian numeric characters
    text = re.sub(r'[٠-٩]', '', text)

    return text
```

انتخاب مدل:

به صورت کلی باید مدلی پیاده سازی شود که بر مبنای جملات قبلی و حتی بعدی آن تصمیم بگیرد، بنابراین داشتن این دو ویژگی ما را محدود به مدل های ترنسفورمرز و مدل های RNN based محدود کرد، در این مدل ها مدل BERT یکی از معروف ترین مدل ها بود برای بررسی، بنابراین از آن شروع کردیم، همچنین در ادامه به مدل ها فارسی tune شده در این زمینه انداختیم که آن هارا استفاده کنیم که به مدل parsBert رسیدیم.

در نهایت توانستیم با مدل XLM به نتیجه دلخواه برسیم.

قابل ذکر هست که به دلیل کمبود سخت افزار موجود مدل large آن خیلی میتوانست بهتر باشد، اما امکان پذیر نبود.

در ادامه توضیحی بر همه روش های موجود داده میشود:

برای تشخیص احساسات متن، روش های مختلفی وجود دارد که از تکنیک های پردازش زبان طبیعی (NLP) استفاده می کنند. برخی از این روش ها عبارتند از:

روش کلمه کلیدی: این روش با استفاده از یک فهرست از کلمات که به هر دسته احساسی مرتبط هستند، احساسات را شناسایی می کند. برای مثال، کلمات مثل "خوشحال"، "لبخند" و "شاد" می توانند به دسته خوشحالی اشاره کنند. این روش ساده و سریع است، اما دقت پایینی دارد و نمی تواند احساسات پیچیده و ضمنی را تشخیص دهد.

روش مبتنی بر واژه نامه: این روش با استفاده از یک واژه نامه از کلمات که به هر دسته احساسی یک امتیاز مثبت یا منفی نسبت داده شده است، احساسات را شناسایی می کند. برای مثال، کلمه "عالی" می تواند یک امتیاز مثبت بالا و کلمه "بد" می تواند یک امتیاز منفی پایین داشته باشد. این روش می تواند احساسات را با در نظر گرفتن شدت آنها تشخیص دهد، اما به واژه نامه های بزرگ و به روز نیاز دارد و نمی تواند احساسات متناقض و بستگی به موضوع را تشخیص دهد.

روش یادگیری ماشین: این روش با استفاده از الگوریتم های یادگیری ماشین مانند Decision Tree، Naive Bayes، SVM و غیره، احساسات را شناسایی می کند. این روش با استفاده از یک مجموعه داده از متن هایی که قبلاً برچسب گذاری شده اند، یک مدل آموزش می دهد و سپس متن های جدید را با استفاده از مدل طبقه بندی می کند. این روش می تواند احساسات را با دقت بالاتری تشخیص دهد، اما به مجموعه داده های کافی و کیفی نیاز دارد و ممکن است به ویژگی های مختص زبان و فرهنگ وابسته باشد.

روش یادگیری عمیق: این روش با استفاده از شبکه های عصبی عمیق مانند CNN، LSTM، GRU و غیره، احساسات را شناسایی می کند. این روش با استفاده از یک مجموعه داده از متن هایی که قبلاً برچسب گذاری شده اند، یک مدل عمیق آموزش می دهد و سپس متن های جدید را با استفاده از مدل طبقه بندی می کند. این روش می تواند احساسات را با دقت بیشتری تشخیص دهد، اما به مجموعه داده های بزرگ و پیچیده و منابع محاسباتی قوی نیاز دارد.

رویکردهای ترکیبی: رویکردهای مبتنی بر قانون و رویکردهای مبتنی بر ML در راه حل جدیدی ادغام می شوند که نقاط قوت هر دورویکرد را دارد و در عین حال ضعف های مرتبط با آنها را کاهش می دهد. ایده این است که می توان با اجرای مجموعه ای از طبقه بندی کننده ها و افزودن اطلاعات زبانی غنی از دانش از فرهنگ لغت، نتایج دقیق تری در تشخیص احساسات به دست آورد.

پیاده سازی:

در اینجا ما روش های مختلفی را مورد بررسی قرار دادیم:

- distilbert-base-uncased
- HooshvareLabbert-base-parsbert-uncased
- michellejeliemotion_text_classifie
- xlm-roberta-base

معرفی مدل های نام برده:

distilbert-base-uncased: این مدل یک نسخه فشرده و سبکتر از مدل BERT است که با استفاده از روش تقطیر دانش (knowledge distillation) آموزش داده شده است. این مدل 40 درصد پارامترهای کمتری نسبت به BERT دارد و 60 درصد سریعتر اجرا می شود، در حالی که بیش از 95 درصد عملکرد BERT را حفظ می کند.

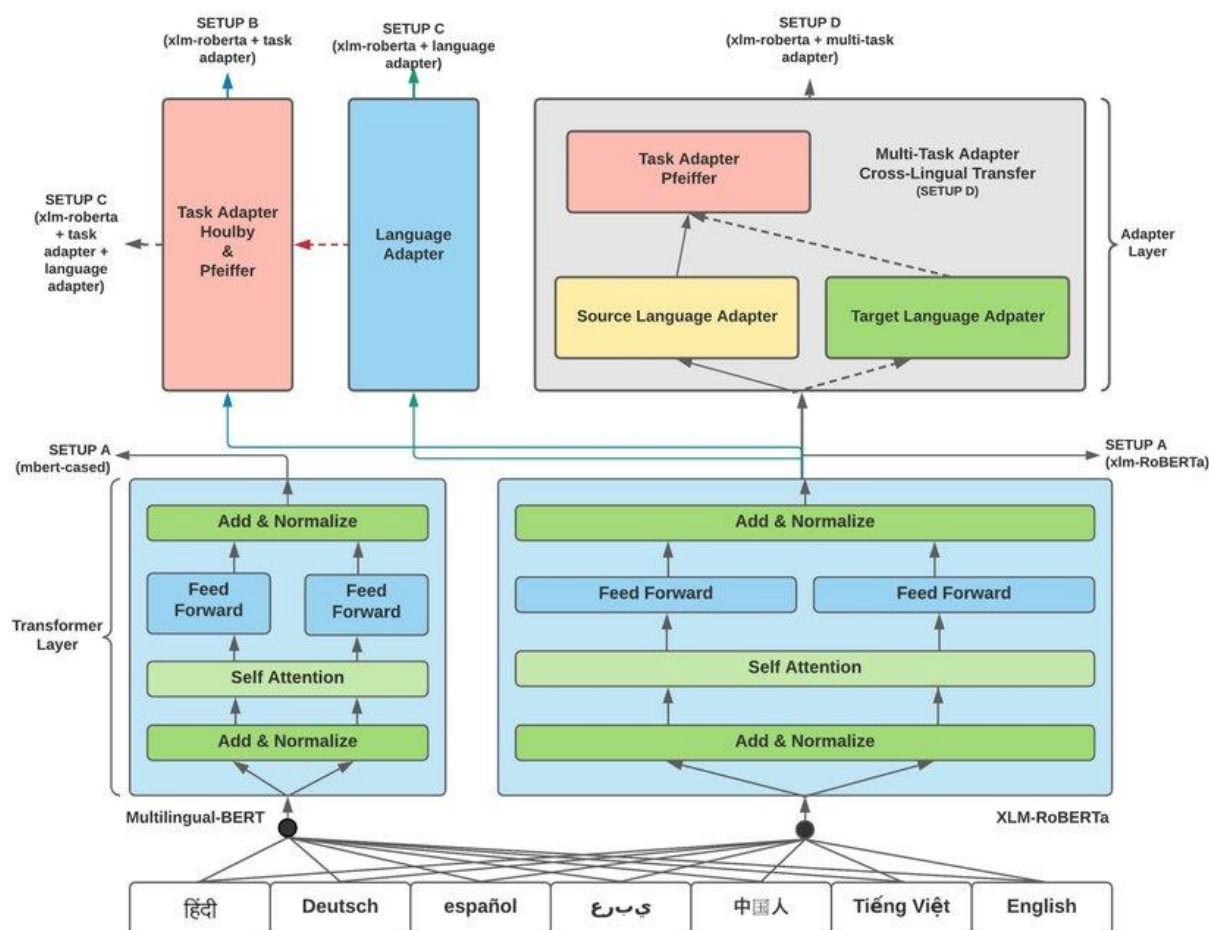
HooshvareLab/bert-base-parsbert-uncased: این مدل یک نسخه از BERT است که بر روی داده های فارسی آموزش داده شده است. این مدل توسط تیم HooshvareLab ایجاد شده است و برای وظایف مختلفی مانند تحلیل احساسات، تشخیص نام های اشخاص، تشخیص موجودیت های نامدار و غیره بهینه سازی شده است

michellejeli/emotion_text_classifier: این مدل یک طبقه بندی کننده متن بر اساس احساسات است که بر روی مجموعه داده EmotionLines آموزش داده شده است. این مدل می تواند احساسات مختلفی مانند شادی، غم، عصبانیت، ترس و غیره را در متون شناسایی کند به این دلیل از این استفاده کردیم که قبلا بر روی احساسات یادگرفته بود و احتمال میدادیم برای تسک مشابه هم مفید باشد.

xlm-roberta-base: این مدل یک نسخه از مدل RoBERTa است که بر روی 100 زبان مختلف آموزش داده شده است. این مدل توسط تیم Facebook AI ایجاد شده است و برای وظایف مختلفی مانند ترجمه ماشینی، تحلیل احساسات، تشخیص زبان و غیره قابل استفاده است.

که بهترین نتیجه را xlm roberta-base داشت بنابراین درباره ان بحث میکنیم:

این مدل از یک شبکه عصبی ترانسفورمر (Transformer) برای یادگیری نمایش‌های زبانی استفاده می‌کند. این شبکه از دو بخش اصلی تشکیل شده است: یک بخش کدگذار (Encoder) که مسئولیت درک متن را دارد و یک بخش کدگشا (Decoder) که مسئولیت تولید متن را دارد. این مدل از دو روش مختلف برای آموزش استفاده می‌کند: یک روش مبتنی بر پیش‌بینی کلمه بعدی (Causal Language Modeling) و یک روش مبتنی بر پیش‌بینی کلمات ماسک شده.



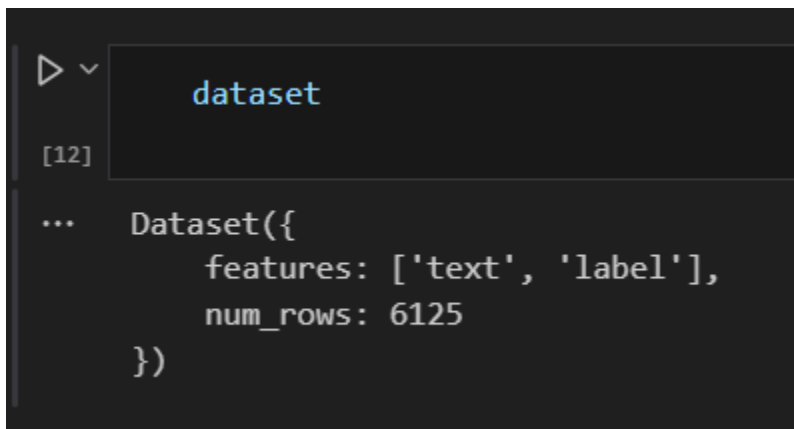
مراحل پیاده سازی:

اول از همه دیتا های آموزش و تست را از سایت داده شده در colab دانلود کردیم.
و با استفاده از read در پاندا آن هارا خواندیم :

```
import pandas as pd
from parsivar import Normalizer
import re

# Read the TSV file into a DataFrame
df = pd.read_csv('train.tsv', sep='\t', header=None, names=['Sentence', 'Label'], encoding='utf-8')
df_test = pd.read_csv('test.tsv', sep='\t', header=None, names=['Sentence', 'Label'], encoding='utf-8')
```

در ادامه مراحل پیش پردازشی که بالاتر گفته شد را بر روی آن انجام دادیم تا به دیتاست رسیدیم که بتوانیم بر روی آن عمل توکنایز کردن را انجام دهیم :



```
[12] dataset

... Dataset({
      features: ['text', 'label'],
      num_rows: 6125
})
```

حال برای توکنایز کردن صرفا توکنایزر مدل را دانلود کرده و با استفاده از تابع زیر بر روی تمامی جملات گذارندیم تا آماده شود:

```
def preprocess_function(examples):
    return tokenizer(examples["text"], truncation=True , padding=True)

dataset_yok = dataset.map(preprocess_function, batched=True)
dataset_test_yok = dataset_test.map(preprocess_function, batched=True)
```

نهایتا دیتا کولر را برای گذاشتن پدینگ قرار دادیم و متریک دقت را هم لود کردیم تا به ترینر دهیم و مدل را با توجه به داده های الآن ترین کنیم:

```

from transformers import DataCollatorWithPadding

data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

```

```

import evaluate

accuracy = evaluate.load("accuracy")

```

Could not render content for 'application/vnd.jupyter.widget-view+json'
{"version_major":2,"version_minor":0,"model_id":"54d99329a5a2427ea4bb59066511c9e7"}

```

import numpy as np

def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    predictions = np.argmax(predictions, axis=1)
    return accuracy.compute(predictions=predictions, references=labels)

```

مدل را لود کرده با 8 هفت لیبل از طریق ریپو آن :

```

from transformers import AutoModelForSequenceClassification, TrainingArguments, Trainer

model = AutoModelForSequenceClassification.from_pretrained(
    "xlm-roberta-base", num_labels=7, id2label=id2label, label2id=label2id
)

```

و در نهایت برای فایل تیون کردن مدل هایپر پارامتر های آن را تنظیم کرده و آموزش را شروع کردیم :

```
training_args = TrainingArguments(  
    output_dir="my_awesome_model",  
    learning_rate=1e-5,  
    per_device_train_batch_size=16,  
    per_device_eval_batch_size=16,  
    num_train_epochs=10,  
    weight_decay=0.05,  
    evaluation_strategy="epoch",  
    save_strategy="epoch",  
    load_best_model_at_end=True,  
)  
  
trainer = Trainer(  
    model=model,  
    args=training_args,  
    train_dataset=dataset_yok,  
    eval_dataset=dataset_test_yok,  
    tokenizer=tokenizer,  
    data_collator=data_collator,  
    compute_metrics=compute_metrics,  
)  
  
trainer.train()
```

برای آموزش و ازمون از نرخ آموزش $1e-5$ و دسته هایی به اندازه 16 (batch_size) با تعداد گام 10 (train_epochs) استفاده کردیم (تعداد ایپوک بالاتر باعث اورفیت شدن میتواندست شود و همچنین بچ های بزرگتر نمیتوانستیم استفاده کنیم زیرا رم پر میشد به علت لود مدل و دیتا همزمان).

برای مدل های متفاوت نتایج متفاوتی بدست امد که به شرح زیر هست:

مدل	تابع ضرر	کمترین دقت	میانگین دقت
distilbert-base-uncased	0.95	0.32	0.37
HooshvareLabbert-base-parsbert-uncased	0.46	0.6	0.65
michellejeliemotion_text_classifie	1.26	0.2	0.65
xlm-roberta-base	0.85	0.37	0.65

[3830/3830 1:39]

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	1.601365	0.371851
2	1.691700	1.186528	0.595135
3	1.104600	1.063597	0.647263
4	0.902900	1.056085	0.660295
5	0.902900	1.012143	0.679409
6	0.765000	0.967625	0.707211
7	0.669600	0.989024	0.698523
8	0.575600	1.021573	0.710686
9	0.575600	1.075202	0.688097
10	0.530200	1.050573	0.695917

نتیجه:

با دیتاست یکسان xlm-roberta-base بهترین نتیجه داشت که به دقت 71 در بهترین مدل رسیدیم. با توجه به اینکه لاس ترین در حال کم شدن هست ولی لاس

ولیدیشن ثابت، میتوانیم بگوییم مدل در حال ارفیت کردن هست و weight decay و معیار های رگولاریزیشن را باید افزایش دهیم و دوباره ترین کنیم که وقتش نشد.

نتیجه تابع های پیاده سازی شده برای این مدل:

Precision: 0.6973679976218039

Recall : 0.6989974153965672

F1-score : 0.6917565569529108

یک نمونه از خروجی:

ورودی: ("چقدر خوبه بیرون رفتن و بازی کردن خیلی قشنگه")

خروجی: 'label': 'HAPPY', 'score': 0.9471352100372314

بابت بخش امتیازی دوتا دیتا ست هم پیدا کردیم که احساسات را داشته باشند و به توان به عنوان ورودی از آن ها استفاده کرد، منتها به ترین مدل و ساخت اون نرسیدیم.

لینک دیتا ست ها :

<https://www.kaggle.com/datasets/parulpandey/emotion-dataset>

<https://huggingface.co/datasets/TrainingDataPro/facial-emotion-recognition-dataset>

منابع:

[2207.11808.pdf \(arxiv.org\)](https://arxiv.org/pdf/2207.11808.pdf)

[arman-text-emotion/dataset at main · Arman-Rayan-Sharif/arman-text-emotion · GitHub](#)

[Hugging Face – The AI community building the future.](#)

[Emotion Classification | Papers With Code](#)

<https://huggingface.co/FacebookAI/xlm-roberta-base>

<https://huggingface.co/MilaNLPProc/xlm-emo-t>