به نام خدا

درس مبانی یادگیری عمیق

گزارش پروژه پایانی

استاد درس : دکتر مرضیه داوودآبادی

دستیاران : مرتضی حاجی‌آبادی، سحر سرکار، فائزه صادقی، مهسا موفق بهروزی، الناز رضایی، پریسا ظفری، حسن حماد، سید محمد موسوی، کمیل فتحی، شایان موسوی نیا، امیررضا ویشته

دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر

نیمسال اول تحصیلی ۱۴۰۲ - ۱۴۰۳

موضوع:

تحلیل احساسات در متن فارسی

ردیف	نام ونام خانوادگی	شماره دانشجویی
۱	بکتاش انصاری	99521082
۲	نوید ابراهیمی	99521001

جدول ۱: مشخصات اعضای گروه

۱ شرح موضوع و مجموعه دادگان

تجزیه و تحلیل احساسات، که به عنوان عقیده کاوی نیز شناخته می شود، یک رشته مطالعاتی است که نظرات، احساسات، تجربیات و عواطف افراد را که از طریق زبان نوشتاری بیان می‌شود، تجزیه و تحلیل می کند. این فیلد زیرمجموعه‌ای از پردازش زبان طبیعی، داده کاوی، وب کاوی و متن کاوی است.
وقتی صحبت از متن فارسی می‌شود، تحلیل احساسات با چالش های منحصر به فردی مواجه است. زبان فارسی ویژگی‌های خاصی دارد و نیازمند روش‌ها و مدل‌های منحصربه‌فردی برای تحلیل احساسات است.

مجموعه داده ArmanEmo مجموعه‌ای از احساسات با برچسب انسانی است که شامل بیش از 7000 جمله فارسی است که برای هفت دسته برچسب‌گذاری شده‌اند. این مجموعه داده توسط میرزایی و همکاران معرفی شده است. در مطالعه خود آن را با عنوان "ArmanEmo: مجموعه داده فارسی برای تشخیص احساسات مبتنی بر متن" معرفی کرده‌اند.

جملات موجود در مجموعه داده ArmanEmo از منابع مختلف از جمله نظرات توییتر، اینستاگرام و دیجی کالا جمع آوری شده است. برچسب‌ها بر اساس شش احساس اصلی (خشم، ترس، شادی، نفرت، غم، شگفتی) و دسته دیگری (سایر) برای در نظر گرفتن هر احساس دیگری که در مدل Ekman گنجانده نشده است.

۲ پیش‌پردازش داده ها

پیاده‌سازیی که در این قسمت انجام دادیم را به دو قسمت تقسیم می‌کنم:
1- نمایش مهم‌ترین اطلاعات دیتاست: در این قسمت با ابزارهای pandas و matplotlib نکاتی از قبیل بیشترین کلمات تکرار شده در متن، تعداد کلمات منحصر به فرد در هر label و 5 کلمه پرکاربرد در هر label را نمایش دادیم. این آمارها در انتهای کار که مدل‌ها را شروع به ارزیابی میکنیم قطعا به کار ما می‌آیند.
2- تمیز کردن دیتاست: در این مرحله از کتابخانه HAZM از شرکت روشن استفاده کردیم. در ابتدا داده‌ها را نرمالایز کردیم. یکی از نمونه‌های اصلاح شده با اینکار حذف فاصله‌های اضافی بین کلمات است. در ادامه از Stemmer استفاده کردیم. با اینکار توانستیم به ریشه اصلی لغات پی ببریم. به طور مثال اگر لغت "کتاب‌ها" وجود داشت، با اینکار این لغت تبدیل به "کتاب" می‌شود. با این تبدیل‌ها می‌توانیم مدل را دقیق‌تر آموزش دهیم.
روشی دیگر که بررسی کردیم lemmatizer بود. در این مرحله با انتخاب هر کلمه، آن را تبدیل به ریشه آن می‌کنیم. ولی با این روش چون کلمه به ریشه‌اش تبدیل می‌شود شاید آن معنای احساسات خود را از دست بدهد. پس از آموزش مدل با این روش و دقت نامطلوب ورودی، این روش را حذف کردیم. در رابطه با متدهای پیش‌پردازش می‌توانیم به سایت اصلی کتابخانه HAZM مراجعه کنید.

برای امبدینگ کلمات مجموعه داده‌هایمان نیز از tokenizer های مدل‌های استفاده شده بهره بردیم.

از 2 مدل برای فرآیند Fine tuning استفاده کردیم:

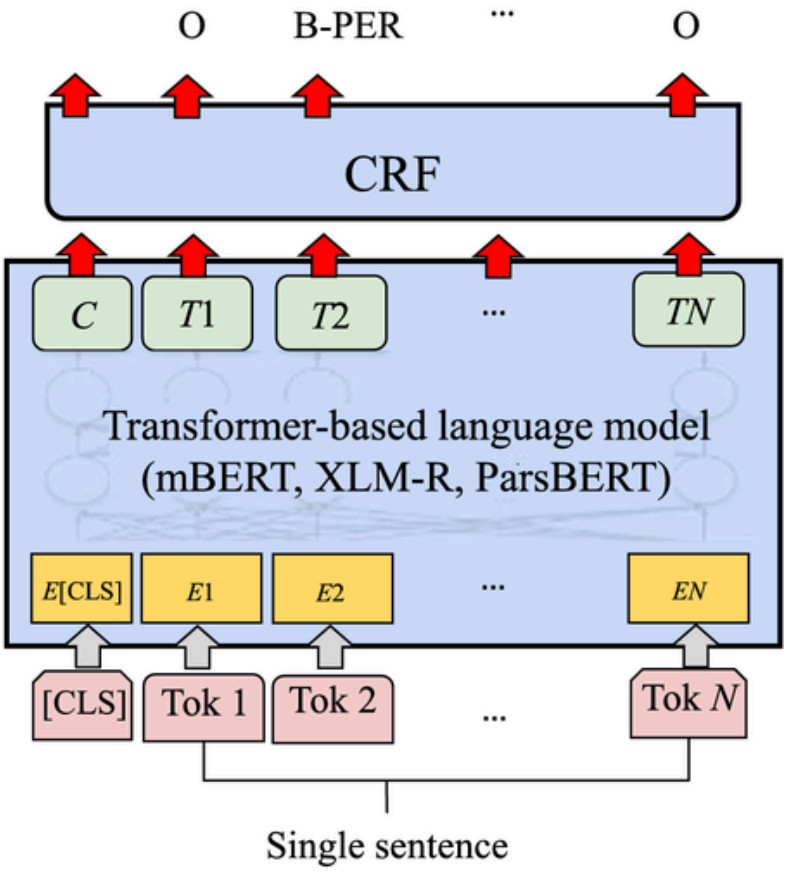
1- Bert:

به عنوان یکی از مدل‌های پایه، ما از یک مدل از پیش آموزش‌دیده برای زبان فارسی، یعنی ParsBERT استفاده کردیم. ParsBERT یک مدل تک زبانه است که بر اساس معماری BERT پیاده‌سازی شده است. طراحان این مدل نشان داده‌اند که مدل ParsBERT از BERT چند زبانه و مدل‌های قبلی در چندین تسک NLP فارسی، از جمله طبقه‌بندی متن و تحلیل احساسات، بهتر عمل می‌کند. در مقایسه با مدل BERT چند زبانه، ParsBERT در مجموعه‌ای بزرگتر و متنوع تر (از نظر دامنه موضوعات و سبک نگارش) مجموعه داده‌های فارسی از پیش آموزش دیده، آموزش دیده است.

2- XLM-RoBERTa:

ما همچنین از مدلی به نام XLM-RoBERTa به عنوان دیگر مدل خود استفاده می کنیم. XLM-RoBERTa یکی دیگر از مدل های transformer-based masked language model است که از قبل روی متونی به ۱۰۰ زبان دنیا آموزش داده شده است. این مدل زبانی چندزبانه منجر به عملکردی پیشرفته در طبقه‌بندی بین زبانی، برچسب‌گذاری دنباله‌ای و پاسخ‌گویی به سؤالات شده است که در معیارهای چندزبانی مختلف از BERT چند زبانه (mBERT) بهتر عمل می‌کند.

اگرچه مشخص است که ParsBERT به عنوان یک مدل زبانی تک زبانه از BERT چند زبانه در کارهای مختلف در زبان فارسی بهتر عمل می کند، اما ما تصمیم گرفتیم عملکرد تغییرات XLM-RoBERTa را در مقابل ParsBERT در تشخیص احساسات نیز مقایسه کنیم.



۴ اقدامات انجام شده

1. **اطلاعات آماری:** ابتدا مجموعه ای از اطلاعات آماری را مانند کلمات پرتکرار و ... که پیش‌تر راجع‌به آنها صحبت کردیم را گزارش کردیم.

2. **پیش‌پردازش:** سپس با استفاده از کتابخانه هضم که درباره آن صحبت کرده بودیم فرآیند پیش‌پردازش و تمیز کردن داده را انجام دادیم.

3. **آموزش مدل PARSBERT**:
ما برای آموزش و ارزیابی مدل های مد نظر از کتابخانه Transformers Hugging Face استفاده کردیم. این کار را با تعریف تابع «compute_metrics» برای محاسبه accuracy, recall، F1-Score و Precisicion شروع کردیم. سپس، نام مدل را مشخص کردیم و Tokenizer و model را بارگذاری کردیم و مجموعه داده‌ها را برای train و test آماده کردیم. هایپرپارامتر های آموزش مانند output directory, number of epochs, batch sizes, learning rate و سایر پارامترها را تعریف کردیم.

ما برای آموزش از کلاس Trainer با آرگومان ها و مجموعه داده های مشخص شده استفاده می‌کنیم. پس از آموزش، مدل را بر روی مجموعه داده test آزمایش می‌کنیم و نتایج ارزیابی را چاپ می‌کنیم. مدل مورد استفاده در اینجا یک مدل BERT از پیش آموزش‌دیده‌شده برای زبان فارسی است و برای یک کار sequence classification با هفت برچسب تنظیم شده است. داده ها با استفاده از همان tokenizer که برای پیش آموزش مدل استفاده شده بود، encode می شوند. مجموعه داده های آموزش و ارزیابی از این رمزگذاری ها و برچسب های مربوطه ایجاد می شوند. فرآیند آموزش توسط کلاس "Trainer" مدیریت می شود که معیارهای مشخص شده را در طول ارزیابی نیز محاسبه می کند. بهترین مدل در پایان آموزش بر اساس evaluation loss ذخیره می شود. نتایج ارزیابی نهایی چاپ می شود.

3.**آموزش مدل XLM-RoBERTa:**
این آموزش را با استفاده کتابخانه pytorch و tqdm انجام دادیم. به این شکل که مانند مدل قبلی tokenizer و model را از hugging face بارگزاری کردیم و با استفاده از DataLoader مجموعه داده test و train را آماده کردیم. و فرآیند اموزش مدل را انجام دادیم. برای آنکه میزان پیشرفت مدل در فرآیند آموزش را مشاهده کنیم از کتابخانه tqdm استفاده کردیم.

4. **ذخیره مدل ها در hugging face:**
برای آنکه مدل‌های آموزش دیده شده را از دست ندهیم، هر دو مدل را بر روی hugging face ذخیره کردیم که از طریق لینک‌های زیر قابل دسترس‌اند و میتوانید هر دو مدل را بصورت انلاین تست کنید:

- ParsBERT
- XLM-ROBERTA

حال برای تست مدل کافیست مدل را به همراه tokenizer از این دو لینک load کنیم.

5. **ارزیابی مدل با استفاده از تابع predict_label:**

در نهایت نیز مدل هارا load میکنیم و با استفاده از تابع predict_label که در انتهای نوت‌بوک قرار دارد مدل هارا بر روی 5 نمونه ارزیابی میکنیم.

نکته: دقت کنید که مراحل آموزش و ارزیابی چندین بار با هایپر پارامترهای مختلف انجام شده است تا به یک نتیجه قابل قبول و خوب برسیم.

۵ ارزیابی مدل

نتایج مدل‌ها بر روی داده مطابق جدول زیر است:

Model/Metric	Precision	Recall	F1	Accuracy
ParsBERT	68%	65%	65%	65%
XLMROBERTA-L	74%	67%	68%	67%

```
texts = ["بشدت از دست علی عصبانی‌ام", "در این زمانه همه به نوعی خود را آرام میکنند", "وای نرسیدم", "ازش متنفرم", "امروز حال من خیلی خوبه"]
predicted_labels = predict_labels(model, tokenizer, texts)
print(predicted_labels)
```

Python

```
['HAPPY', 'HATE', 'FEAR', 'OTHER', 'ANGRY']
```

۶ بخش امتیازی

برای بخش امتیازی ابتدا یک مجموعه داده‌ای را که برای تحلیل احساسات مثبت و منفی و خنثی است را پیدا کردیم. که این داده در این لینک قابل مشاهده است. این داده با نام MVSA شامل مجموعه عکس‌ها و متونی است که از شبکه اجتماعی توئیتر (X) جمع آوری شده است به این شکل که هر sample این مجموعه داده شامل یک عکس و یک توئیت مربوط به آن است و لیبل مربوط به آن‌ها نیز با سه مقدار neutral و positive و negative قابل نمایش است. این دیتاست را درون یک نوت‌بوک جدا load کردیم و سمپلی از آن‌را نیز نمایش دادیم. برای مدل آموزش نیز از ترکیبی از دو مدل Resnet50 و BERT استفاده کردیم. به این صورت که برای مدالیتی عکس، ورودی عکس را به مدل Resnet دادیم و لایه آخر این مدل را حذف کردیم تا به یک feature vector برسیم. برای داده متنی نیز ورودی متن را به مدل BERT دادیم تا یک feature vector نیز از آن استخراج کنیم. در ادامه با استفاده از ابزارهای pytorch این دو vector را به هم concat کردیم تا یک feature vector یکتا داشته باشیم و این وکتور را به عنوان ورودی به یک مدل MLP چهار لایه برای آموزش وارد کردیم و در نهایت مدل را آموزش دادیم. در انتها نیز مدل را بر روی سمپلی از داده تست کردیم.
دقت و loss مدل بر روی داده train و test قابل مشاهده است:

```
Training: 100%|██████████| 13/13 [00:00<00:00, 15.36it/s]
Validation: 100%|██████████| 4/4 [00:00<00:00,  7.77it/s]
Epoch 1/5, Train Loss: 1.1110, Val Loss: 1.1087, Val Accuracy: 0.2200
Training: 100%|██████████| 13/13 [00:01<00:00,  6.59it/s]
Validation: 100%|██████████| 4/4 [00:00<00:00, 17.84it/s]
Epoch 2/5, Train Loss: 1.1039, Val Loss: 1.1017, Val Accuracy: 0.2200
Training: 100%|██████████| 13/13 [00:01<00:00, 11.62it/s]
Validation: 100%|██████████| 4/4 [00:00<00:00, 10.55it/s]
Epoch 3/5, Train Loss: 1.0979, Val Loss: 1.0950, Val Accuracy: 0.5000
Training: 100%|██████████| 13/13 [00:01<00:00, 10.67it/s]
Validation: 100%|██████████| 4/4 [00:00<00:00,  5.22it/s]
Epoch 4/5, Train Loss: 1.0923, Val Loss: 1.0893, Val Accuracy: 0.5000
Training: 100%|██████████| 13/13 [00:01<00:00,  6.98it/s]
Validation: 100%|██████████| 4/4 [00:00<00:00,  6.62it/s]
Epoch 5/5, Train Loss: 1.0872, Val Loss: 1.0840, Val Accuracy: 0.5000
```

```
Label :  positive
Nothin but the pastel sky hangin over you and I #theOriginalGradient #cottoncandy #sky #bright #happiness
```

```
model.eval()
feature_vector = torch.tensor(sample["featureVector"]).view(-1, input_size).to(device)
output = model(feature_vector.float())
pred = output.argmax(dim=1).tolist()
label_map[pred[0]]
```

```
'positive'
```

- https://arxiv.org/abs/2207.11808 [ArmanEmo]
- https://arxiv.org/abs/2005.12515 [ParsBert]
- https://arxiv.org/pdf/1911.02116.pdf [XLM-ROBERTA]
- https://www.roshan-ai.ir/hazm/docs/ [HAZM]
- https://mcrlab.net/research/mvsa-sentiment-analysis-on-multi-view-social-data/ [MVSA]
- https://arxiv.org/abs/1512.03385 [Resnet]
- https://arxiv.org/abs/1810.04805 [BERT]