تمرین چهارم NLP

سوال 1

تفاوت اصلی بین رویکردهای سنتی NER و GPT-NER به شرح زیر است:

رویکردهای NER اکثرا در برچسب گذاری Sequence تمرکز دارند جایی که هر کلمه در یک جمله برچسب خاصی دریافت می کند که نشان می دهد آیا نشان دهنده یک موجودیت است یا خیر. این فرآیند برچسب گذاری بیشتر به مدل های آماری یا شبکه های عصبی متکی است که برای عملکرد مؤثر به داده های آموزشی برچسب گذاری شده گسترده نیاز دارند.
ولی در مقابل GPT-NER این وظیفه را به یک مسئله تولید متن تبدیل می کند و به جای برچسب گذاری هر توکن GPT-NER نسخه ای اصلاح شده از متن ورودی تولید می کند که در آن موجودیت ها با توکن های ویژه مشخص شده اند. این تغییر به GPT-NER اجازه می دهد تا از نقاط قوت مدل های زبانی بزرگ (LLMs) بهره برداری کند و آن را به ویژه در محیط های با منابع کم، سازگارتر و مثرتر کند.

روش های سنتی NER شامل موارد زیر است:
برچسب گذاری هر کلمه: به هر کلمه برچسبی اختصاص داده می شود که نقش آن را نشان می دهد. به عنوان مثال در جمله "باراک اوباما رئیس جمهور بود" عبارت "باراک اوباما" به عنوان یک شخص برچسب گذاری می شود.
استفاده از مدل های آماری: روش های قبلی به مدل های آماری متکی بودند که نیاز به مهندسی ویژگی های دقیق داشتند که در واقع ایجاد قوانین درباره اینکه چه چیزی بر اساس دانش قبلی یک موجودیت را تشکیل می دهد.
تکنیک های یادگیری عمیق: روش های جدیدتر از شبکه های عصبی برای یادگیری خودکار ویژگی ها از داده ها استفاده می کنند. این مدل ها می توانند الگوها را در دیتاست های بزرگ شناسایی کنند اما هنوز به شدت به داشتن تعداد زیادی دیتای برچسب گذاری شده برای آموزش وابسته هستند.

روش های GPT-NER به شرح زیر است:
تمرکز بر تولید متن: به جای برچسب گذاری ساده کلمات GPT-NER این وظیفه را به عنوان تولید متنی که در آن موجودیت ها به شیوه ای خاص برجسته شده اند انجام می دهد.
روش Self-Verification: برای بهبود دقت و کاهش خطاها GPT-NER شامل مرحله ای است که در

آن خروجی های خود را بررسی می کند. این کمک می کند تا اطمینان حاصل شود که آنچه به عنوان یک موجودیت شناسایی می کند واقعاً با دسته بندی های مورد انتظار مطابقت دارد.
یادگیری از نمونه های کم: یکی از ویژگی های برجسته GPT-NER توانایی آن در یادگیری از تنها چند مثال است. این بدان معناست که می تواند به سرعت به وظایف جدید سازگار شود بدون اینکه به داده های آموزشی گسترده نیاز داشته باشد، که آن را در موقعیت های مختلف انعطاف پذیرتر می کند.

B) GPT-NER وظیفه برچسب گذاری sequence های NER را به یک وظیفه تولید متن تبدیل می کند و نحوه شناسایی موجودیت ها در متن را دوباره فرمت می کند. این رویکرد نوآورانه از قابلیت های مدل های زبانی بزرگ (LLMs) بهره می برد تا خروجی هایی تولید کند که به جای برچسب گذاری ساده موجودیت ها را برجسته کنند.

حالا این روش چگونه GPT-NER کار می کند؟
در NER سنتی به هر کلمه در یک جمله برچسبی اختصاص داده می شود که نشان می دهد آیا بخشی از یک موجودیت است و نوع آن چیست ولی GPT-NER این وظیفه را به صورت زیر فرمول بندی می کند :
تولید متن به جای برچسب گذاری: به جای برچسب گذاری هر توکن GPT-NER نسخه ای تغییر یافته از متن ورودی تولید می کند که در آن موجودیت ها با توکن های ویژه علامت گذاری شده اند. به عنوان مثال ورودی "تهران یک شهر است" به "@@ تهران ## یک شهر است" تبدیل می شود. در اینجا نشانه های "@@" و "##" نشان می دهند که "تهران" یک موجودیت است .
فرمت خروجی ساده شده: این تغییر کار را برای مدل ساده تر می کند. فقط نیاز دارد که مکان موجودیت ها را شناسایی کند و سپس بقیه جمله را به همان صورت تکرار کند. این پیچیدگی را در مقایسه با تطبیق برچسب ها با توکن های فردی کاهش می دهد.
روش Self-Verification: این روش را بالاتر توضیح دادیم که برای رفع نادرستی های احتمالی GPT-NER شامل یک مرحله Self-Verification است که در آن مدل بررسی می کند آیا موجودیت های شناسایی شده با دسته بندی های مورد انتظار مطابقت دارند یا خیر.
مثال برای نشان دادن این تغییر، به مثال زیر توجه کنید :
جمله ورودی: "تهران یک شهر است".
خروجی سنتی NER :
"تهران" ← مکان
"است" ← غیر موجودیت

"یک" ← O
"شهر" ← O

خروجی GPT-NER :

"@@ تهران ## یک شهر است"

در این مورد GPT-NER به طور موثری تهران را به عنوان یک موجودیت علامت گذاری می کند در حالی که بقیه جمله دست نخورده باقی می ماند، که این امر پردازش و درک آسان تر توسط LLM ها را ممکن می سازد. این روش نه تنها عملکرد را بهبود می بخشد بلکه قابلیت سازگاری را در سناریوهایی با داده های آموزشی محدود نیز افزایش می دهد که GPT-NER را در کاربردهای دنیای واقعی که نمونه های برچسب گذاری شده ممکن است کم باشند به ویژه ارزشمند می سازد.

C) استراتژی Self-Verification در GPT-NER یک عنصر حیاتی است که برای بهبود دقت شناسایی موجودیت های نام دار (NER) طراحی شده است و به مشکل توهم می پردازد، جایی که مدل ها به اشتباه موجودیت های غیر را به عنوان موجودیت برچسب گذاری می کنند. در اینجا توضیحی ساده از نقش و اهمیت آن آورده شده است:

استراتژی Self-Verification مدل را مجبور می کند تا پس از شناسایی موجودیت ها خروجی های خود را ارزیابی کند. در واقع از مدل می خواهد بررسی کند که آیا موجودیت های استخراج شده واقعا به هر دسته بندی از پیش تعریف شده تعلق دارند یا خیر. این روش کمک می کند:
کاهش اعتماد به نفس بیش از حد: مدل های زبانی بزرگ (LLM) گاهی اوقات در پیش بینی های خود بیش از حد مطمئن هستند که منجر به اشتباهات می شود. با ترغیب مدل به تأیید خروجی هایش، احتمال کمتری دارد که برچسب های نادرست را تأیید کند.
افزایش دقت: این استراتژی به مدل اجازه می دهد تا فرآیند شناسایی موجودیت خود را بهبود بخشد و در نهایت عملکرد کلی وظایف NER را بهبود بخشد.
پرداختن به مسائل توهم: توهمات زمانی رخ می دهند که یک مدل اطلاعاتی قابل قبول اما نادرست تولید کند. مرحله Self-Verification به کاهش این مشکل کمک می کند و اطمینان حاصل می کند که فقط موجودیت های معتبر شناسایی شوند.
به طور خلاصه، استراتژی Self-Verification در GPT-NER نه تنها دقت را بهبود می بخشد بلکه با بررسی فعال صحت خروجی های خود، نقش کلیدی در اطمینان از اینکه شناسایی موجودیت های مدل هم قابل اعتماد و هم معتبر باشد، ایفا می کند.

ساختار پرامپت برای GPT-NER شامل سه جزء اصلی است :
توضیحات Task: این بخش نمای کلی از آنچه مدل انتظار می رود انجام دهد را ارائه می دهد.
Few-shot: این مؤلفه شامل مثال هایی است که مدل را در نحوه فرمت بندی خروجی اش راهنمایی می کند.
جمله ورودی: این بخش شامل جمله واقعی است که باید موجودیت ها در آن شناسایی شوند.

سوال 2

A) تعریف مدل های Generative و Discriminative

مدل های Generative: این مدل ها توزیع احتمال مشترک p(d,c) داده ها d و برچسب های کلاس c را یاد می گیرند. با مدل سازی نحوه تولید داده ها می توانند نمونه های جدیدی از داده ها تولید کنند. مدل های Generative می توانند برای وظایفی مانند classification، تخمین چگالی، و تولید نمونه های جدید از توزیع های یادگرفته شده استفاده شوند. نمونه هایی شامل بیز ساده و مدل های مارکوف پنهان می شوند.

مدل های Discriminative: در مدل های Discriminative توزیع احتمال شرطی p(c|d) را یاد می گیرند که نمایانگر احتمال برچسب های کلاس با توجه به داده ها است. آن ها بر مدل سازی مرز تصمیم گیری بین کلاس ها تمرکز می کنند نه توزیع زیرین داده ها. مدل های تفکیکی اغلب برای وظایف classification موثرتر هستند زیرا به طور مستقیم رابطه بین ویژگی ها و برچسب های کلاس را مدل سازی می کنند. نمونه ها شامل Logistic Regression، ماشین های بردار پشتیبان(SVM) ، و میدان های تصادفی شرطی می باشند (CRF).

ویژگی	مدل های Generative	مدل های Discriminative
تمرکز	نحوه تولید داده ها	مرزهای تصمیم گیری بین کلاس ها
تولید داده	می تواند نمونه های جدید تولید کند	نمی تواند نمونه های جدید ایجاد کند
پیچیدگی	به طور کلی به دلیل مدل سازی مشترک پیچیده تر است	ساده تر، با تمرکز فقط بر classification
عملکرد در classification	اغلب برای وظایف طبقه بندی کمتر مؤثر است	به طور کلی در وظایف طبقه بندی از مدل های generative بهتر عمل می کنند.
نمونه ها	Naive Bayes, Hidden Markov Models	Logistic Regression, Support Vector Machines

B) مدل های Maxent به دلایل زیر اکثرا بر مدل های بیز ساده برای وظایف classification ترجیح داده می شوند:

انعطاف پذیری ویژگی ها: Maxent اجازه استفاده از ویژگی های مختلف از جمله ویژگی های پیچیده ای که روابط موجود در داده ها را به تصویر می کشند را می دهد که این انعطاف پذیری به آن کمک می کند تا بهتر به زمینه های مختلف به ویژه در پردازش زبان طبیعی سازگار شود.

رویکرد Discriminative: Maxent به طور مستقیم احتمال شرطی برچسب های کلاس را با توجه به داده ها مدل سازی می کند p(c|d) که به این معنی است که بر تمایز بین کلاس ها تمرکز دارد. این کار اغلب منجر به دقت بهتر در classification نسبت به Naive Bayes می شود که توزیع مشترک p(d,c) را یاد می گیرد و به فرضیات استقلال قوی بین ویژگی ها تکیه می کند.

مدیریت تعاملات ویژگی ها: Maxent می تواند تعاملات بین ویژگی ها را در نظر بگیرد و به این ترتیب الگوهای دقیق تری را در داده ها شناسایی کند. در مقابل Naive Bayes فرض می کند که ویژگی ها مستقل هستند که می تواند مدل را بیش از حد ساده کند و کارایی آن را کاهش دهد.

Regularization: مدل های ماکسنت معمولا تکنیک های Regularization را شامل می شوند که به جلوگیری از overfitting کمک می کنند و آن ها را در سناریوهای مختلف مقاوم تر می سازند. Naive Bayes این Regularization ذاتی را ندارد، که ممکن است منجر به عملکرد ضعیف تر با داده های پر سر و صدا شود.

C) مدل Maxent به طور گسترده ای در پردازش زبان طبیعی (NLP) برای وظایفی مانند طبقه بندی متن، pos tagging و NER استفاده می شود. یک مثال قابل توجه در دنیای واقعی کاربرد آن در Sentiment Analysis است جایی که هدف طبقه بندی متن (مانند نظرات محصولات یا پست های شبکه های اجتماعی) به عنوان ابراز احساسات مثبت، منفی یا خنثی است.

در Sentiment Analysis یک مدل Maxent می تواند بر روی داده های برچسب گذاری شده آموزش ببیند که در آن هر نمونه متنی با یک برچسب احساس مرتبط است. مدل یاد می گیرد که احساسات را بر اساس ویژگی های مختلف استخراج شده از متن پیش بینی کند.

ساخت ویژگی های مناسب:

برای ساخت ویژگی ها برای یک مدل Maxent در Sentiment Analysis می توان مراحل زیر را دنبال کرد:

1- شناسایی ویژگی های مرتبط: ویژگی ها باید جنبه های مهم متن را که با احساسات همبستگی دارند ثبت کنند. انواع رایج ویژگی ها شامل:

حضور کلمات: توابع شاخص برای حضور کلمات خاص

Nگرام ها: استفاده از یونی گرام ها (کلمات تکی) بای گرام ها (جفت های کلمات متوالی) یا ترای

گرم ها (سه تایی های کلمات متوالی) برای ضبط زمینه.
Sentiment Lexicons: ویژگی هایی که نشان می دهند آیا کلمات به واژه نامه های احساسی مثبت یا منفی از پیش تعیین شده تعلق دارند یا خیر.
Part-of-Speech Tags: گنجاندن اطلاعات دستوری مانند اینکه آیا یک کلمه صفت است یا قید که اغلب احساسات را منتقل می کنند.

2- وزن دهی ویژگی ها: هر ویژگی (c,d) یک وزن λ_i اختصاص داده می شود. مدل Maxent این وزن ها را در طول آموزش یاد خواهد گرفت تا احتمال شرطی داده های مشاهده شده با توجه به ویژگی ها را به حداکثر برساند.
3- آموزش مدل: مدل با استفاده از یک مجموعه داده که هر نمونه شامل یک متن و برچسب احساسات مربوط به آن است آموزش داده می شود. فرآیند آموزش شامل تنظیم وزن ها به گونه ای است که تعداد مورد انتظار ویژگی ها با تعداد تجربی آن ها در داده های آموزشی مطابقت داشته باشد.
4- پیش بینی: برای یک نمونه متنی جدید مدل امتیازهایی را بر اساس ویژگی های فعال و وزن های آن ها محاسبه می کند تا کلاس احساسات محتمل تر را تعیین کند.

D) استفاده از وزن دهی مناسب برای ویژگی ها در مدل های Maxent برای بهبود عملکرد مدل بسیار حیاتی است به ویژه در وظایفی مانند پردازش زبان طبیعی.در اینجا توضیحی مختصر از اهمیت آن ها میاوریم:

اهمیت این کار: وزن های ویژگی نشان می دهند که هر ویژگی چقدر در پیش بینی نتیجه مؤثر است. وزن بالاتر به این معنی است که این ویژگی در تعیین برچسب کلاس، تاثیر بیشتری دارد. این به مدل کمک می کند تا تصمیمات بهتری بر اساس اطلاعات مرتبط بگیرد.
توازن در Contributions: ویژگی های مختلف می توانند سطوح متفاوتی از ارتباط را داشته باشند. وزن هایی که به خوبی تنظیم شده اند اطمینان می دهند که ویژگی های مهم تأثیر بیشتری دارند و در عین حال تاثیر ویژگی هایی که کمتر مرتبط هستند کاهش می یابد. این تعادل مانع از گمراه شدن مدل توسط داده های نامربوط می شود.
بهبود پیش بینی ها: وقتی وزن ها به درستی تنظیم شوند مدل پیش بینی های خود را با مشاهدات واقعی هماهنگ می کند و دقت را افزایش می دهد. این مدل یاد می گیرد الگوهایی را که واقعا برای طبقه بندی مهم هستند شناسایی کند که منجر به نتایج قابل اعتمادتر می شود .

تأثیر بر دقت مدل:
استفاده از وزن های مناسب برای ویژگی ها به طور مستقیم دقت را بهبود می بخشد که به شرح زیر است:
مدل می تواند تمایزات دقیق تری بین کلاس ها ایجاد کند که در وظایف پیچیده ای مانند تحلیل احساسات یا شناسایی موجودیت ها ضروری است.
با تمرکز بر ویژگی های مرتبط مدل از تعصباتی که می تواند ناشی از اطلاعات گمراه کننده یا اضافی باشد، جلوگیری می کند.
ویژگی هایی با وزن مناسب به مدل اجازه می دهند تا به زمینه ها یا دیتاست مختلف سازگار شود و عملکرد آن را در برنامه های مختلف بهبود بخشد .

سوال 3

A) NER یک ساب تسک مهم در پردازش زبان طبیعی (NLP) است که در شناسایی و طبقه بندی موجودیت های کلیدی در متن به دسته های از پیش تعیین شده تمرکز دارد. این دسته ها معمولاً شامل افراد، سازمان ها، مکان ها و اطلاعات متفرقه ای می شوند. هدف اصلی NER استخراج اطلاعات ساختاریافته از متن غیرساختاریافته است که امکان درک و پردازش بهتر داده ها را فراهم می کند.

چگونه NER کار می کند؟

سیستم های NER با تحلیل متن برای شناسایی دنباله های کلماتی که به موجودیت های نام دار مربوط می شوند، عمل می کنند. سپس هر موجودیت شناسایی شده بر اساس نوع آن برچسب گذاری می شود. به عنوان مثال جمله "Meg Whitman, CEO of eBay, said in New York" را سیستم برچسب گذاری می کند:

"Meg Whitman" به عنوان PERSON

"eBay" به عنوان ORGANIZATION

"New York" به عنوان LOCATION

این فرآیند معمولاً شامل چندین مرحله است:

Tokenization: تقسیم متن به واحدهای کوچکتر (tokens).

Entity Recognition: شناسایی اینکه کدام توکن ها به موجودیت های نام دار مربوط می شوند.

Classification: اختصاص هر موجودیت به دسته بندی مربوطه خود.

NER می تواند با استفاده از روش های مختلف پیاده سازی شود از جمله:

سیستم های مبتنی بر قاعده: استفاده از الگوها و قواعد از پیش تعیین شده.

مدل های یادگیری ماشین: استفاده از الگوریتم هایی مانند Hidden Markov Models (HMM) میدان های تصادفی شرطی (CRF) و یا شبکه های عصبی.

رویکردهای ترکیبی: ترکیب تکنیک های مبتنی بر قوانین و یادگیری ماشین برای بهبود دقت.

کاربردهای NER:

NER در حوزه های مختلف کاربردهای گسترده ای دارد:

بازیابی اطلاعات: بهبود موتورهای جستجو با اجازه دادن به آن ها برای درک بهتر پرسش های کاربران و بازیابی اسناد مرتبط.

ترجمه ماشینی: بهبود کیفیت ترجمه با شناسایی و ترجمه دقیق موجودیت های نام دار.

سیستم های پاسخگویی به سوالات: تسهیل سیستم هایی که بر اساس پرسش های خاص پاسخ هایی ارائه می دهند با شناسایی موجودیت های مرتبط در متون.

خلاصه سازی خودکار: استخراج اطلاعات کلیدی از متون بزرگ برای ایجاد خلاصه های مختصر در حالی که جزئیات اساسی حفظ می شوند.

نظارت بر شبکه های اجتماعی: تحلیل احساسات عمومی نسبت به برندها یا افراد با ردیابی ذکر نام موجودیت ها در پلتفرم های مختلف.

B) برای برچسب گذاری جمله «Ali studies at IUST in Tehran» با استفاده از طرح برچسب گذاری IOB (Inside, Outside, Begin)، موجودیت های نام دار را شناسایی کرده و برچسب های مناسب را اختصاص می دهیم که عبارت است از:
Ali: این نام یک شخص است، بنابراین برچسب B-PER را می گیرد (Begin Person) .

studies: این یک فعل رایج است و نمایانگر یک موجودیت نام دار نیست، بنابراین برچسب O را دریافت می کند (Outside) .
at: این یک حرف اضافه است و همچنین نمایانگر یک موجودیت نام دار نیست، بنابراین برچسب O را دریافت می کند.
IUST: این یک اختصار برای یک سازمان (دانشگاه علم و صنعت ایران) است بنابراین برچسب B-ORGرا دریافت می کند.(Begin Organization) .
in: این یک حرف اضافه است و نمایانگر یک موجودیت نام دار نیست، بنابراین برچسب O را دریافت می کند.
Tehran: این نام یک شهر است بنابراین برچسب B-LOC را دریافت می کند Begin) .
Location)

Ali: B-PER

Studies: O

At: O

IUST: B-ORG

in O:

Tehran: B-LOC

D) مدل MEMMs و مدل CRFs هر دو برای وظایف برچسب گذاری sequence استفاده می شوند، اما ویژگی های متمایزی دارند:

ساختار مدل:

- **مدل های MEMM:** آن ها مدل های Maxent را با مدل های مارکوف ترکیب می کنند و پیش بینی ها را بر اساس مشاهده فعلی و برچسب قبلی انجام می دهند. این می تواند منجر به مشکلاتی شود که در آن اشتباهات اولیه بر تصمیمات بعدی تأثیر می گذارند.
- **مدل های CRFs:** این نوع مدل های گرافیکی بدون جهت هستند که کل Sequence برچسب ها را برای پیش بینی در نظر می گیرند و امکان مدیریت بهتر وابستگی ها بین برچسب ها را فراهم می کنند.

تصمیم گیری برای برچسب گذاری:

- **مدل های MEMM:** تصمیم ها به صورت محلی گرفته می شوند که اگر پیش بینی های قبلی نادرست باشند، می تواند باعث ایجاد سوگیری شود.
- **مدل های CRFs:**آن ها کل توالی برچسب را به طور جهانی بهینه سازی می کنند و تاثیر خطاهای محلی را کاهش می دهند .

استنتاج و آموزش:

- **مدل های MEMM:** از روش های استنتاج ساده تری مانند جستجوی حریصانه استفاده کنید که ممکن است همیشه بهترین نتایج را به همراه نداشته باشد.
- **مدل های CRFs:** از تکنیک های پیچیده تری مانند برنامه نویسی پویا (الگوریتم Viterbi) استفاده کنید تا دنباله برچسب ها دقیق تر باشد.

پیچیدگی:

- **مدل های MEMM:** پیاده سازی آن ها آسان تر است اما به دلیل تمرکز محلی شان ممکن است با عملکرد مشکل داشته باشند.
- **مدل های CRFs:** پیچیده تر و محاسباتی تر هستند اما به طور کلی دقت بهتری در وظایفی که نیاز به درک تعاملات برچسب ها دارند، ارائه می دهند .

E) مدل های شناسایی موجودیت های نام دار (NER) نقش مهمی در بهبود سیستم های بازیابی اطلاعات (IR) ایفا می کنند و نحوه استخراج و فهرست بندی اطلاعات از داده های غیر ساختار یافته را بهبود می بخشند که روش های استفاده NER در IR به شرح زیر است:

استخراج اطلاعات: NER به شناسایی و طبقه بندی موجودیت هایی مانند افراد، سازمان ها و مکان ها درون اسناد کمک می کند. این اطلاعات ساختاریافته می تواند برای درک محتوا و زمینه داده ها حیاتی باشد و بازیابی اطلاعات مرتبط بر اساس درخواست های کاربران را آسان تر کند.

ایندکس گذاری: با برچسب گذاری موجودیت های نام دار، سیستم های IR می توانند ایندکس های اطلاعاتی تری ایجاد کنند. به عنوان مثال، اگر یک سند به "Apple Inc." یا "New York" اشاره کند، این موجودیت ها می توانند به طور جداگانه فهرست بندی شوند، که به کاربران این امکان را می دهد که به طور موثرتری به جستجوی شرکت ها یا مکان های خاص بپردازند.

Query Enhancement : NER می تواند با شناسایی موجودیت های نام دار در جستجوهای کاربران آن ها را بهبود بخشد. وقتی کاربری به دنبال "events in Paris" می گردد، سیستم می تواند "Paris" را به عنوان یک مکان شناسایی کرده و مدارکی را که این موجودیت را ذکر می کنند، در اولویت قرار دهد و به این ترتیب مرتبط بودن نتایج جستجو را بهبود بخشد.

جستجوی معنایی: با استفاده از NER سیستم های IR می توانند قابلیت های جستجوی معنایی را پیاده سازی کنند که به آن ها اجازه می دهد روابط بین موجودیت ها را درک کنند. به عنوان مثال، اگر یک سند به "Barack Obama" در ارتباط با "healthcare" بپردازد، سیستم می تواند اسنادی را که به سیاست های healthcare مرتبط با او پرداخته اند، بازیابی کند، حتی اگر آن اصطلاحات خاص به طور صریح ذکر نشده باشند.

Faceted Search: NER امکان گزینه های جستجوی چندوجهی را فراهم می کند که در آن کاربران می توانند نتایج را بر اساس موجودیت های شناسایی شده (مثلاً فیلتر کردن بر اساس شخص یا سازمان) فیلتر کنند و بدین ترتیب تجربه جستجوی دقیق تری را ارائه دهند.

سوال 4

این یک تسک multi-label classification است که دیتاست آن نظرات کاربران دیجی کالا روی محصولات است که قرار است با مدل های RNN, GRU, LSTM آموزش دهیم.

اول دیتاست مربوطه را پیش پردازش می کنیم که شامل کار های زیر است:

- حذف علائم نگارشی
- تبدیل کلمات به فرم پایه (lemmatization,stemming) به کمک کتابخانه ی Hazm
- حذف کلمات stopwords

سپس واژه ها بر اساس **بیشترین تکرار** مرتب می کنیم و فقط واژه های پرتکرار را نگه می داریم و واژه ها به اندیس های عددی نگاشت می شوند.

در ادامه هر جمله به sequence های از اندیس های عددی متناظر با واژه ها تبدیل می شوند.

معماری مدل ها:

لایه ی Embeddding: این لایه کلمات عددی را به بردارهای متراکم (Dense vectors) در فضای چندبعدی نگاشت می دهد.

لایه ی بازگشتی (RNN , GRU , LSTM): sequence ها را تحلیل کرده و وابستگی های زمانی بین کلمات را استخراج می کند.

- **RNN:** ساده ترین نوع لایه بازگشتی که فقط وابستگی کوتاه مدت را یاد می گیرد.
- **GRU:** نوع پیشرفته تر که فراموشی و به خاطر سپردن اطلاعات را بهبود می بخشد.
- **LSTM:** پیشرفته ترین نوع که می تواند وابستگی های بلندمدت بین کلمات را یاد بگیرد

لایه ی Fully Connected: لایه ی پایانی که multi-label regression را تولید می کند.

Dropout: برای جلوگیری از overfitting این لایه برخی نورون ها را به صورت تصادفی غیرفعال می کند.

ابعاد Embeddding بردار کلمات: مقدار ۱۲۸ انتخاب شده است. افزایش این مقدار ممکن است دقت را بیشتر کند اما به هزینه حافظه اضافه می شود.

ابعاد لایه Hidden در RNN: مقدار ۶۴ که تعیین می کند مدل در هر گام زمانی چند ویژگی را یاد بگیرد.

بهینه ساز (Optimizer): از **AdamW** استفاده شده که یادگیری مدل را پایدار و موثر می کند.

دقت مدل SimpleRNN

```
Training SimpleRNN...
100%|██████████| 2802/2802 [00:12<00:00, 220.73it/s]
Epoch 1/10, Train Loss: 0.09661210470066421, Train F1: 0.37141615424365887
Epoch 1/10, Val Loss: 0.07439315161493723, Val F1: 0.4610352055886427
100%|██████████| 2802/2802 [00:12<00:00, 222.20it/s]
Epoch 2/10, Train Loss: 0.07305584182972082, Train F1: 0.4880067431145138
Epoch 2/10, Val Loss: 0.06736496037746462, Val F1: 0.5065407925755172
100%|██████████| 2802/2802 [00:12<00:00, 221.88it/s]
Epoch 3/10, Train Loss: 0.06682406344998104, Train F1: 0.54600091491648
Epoch 3/10, Val Loss: 0.06474280892771678, Val F1: 0.5205499527654932
100%|██████████| 2802/2802 [00:12<00:00, 220.59it/s]
Epoch 4/10, Train Loss: 0.06283673863235774, Train F1: 0.6092341146437168
Epoch 4/10, Val Loss: 0.06303244833429257, Val F1: 0.5886910348158076
100%|██████████| 2802/2802 [00:12<00:00, 221.60it/s]
Epoch 5/10, Train Loss: 0.059995291502661016, Train F1: 0.6418802931192139
Epoch 5/10, Val Loss: 0.062121736305565195, Val F1: 0.6462513593620295
100%|██████████| 2802/2802 [00:13<00:00, 212.21it/s]
Epoch 6/10, Train Loss: 0.058513152214308285, Train F1: 0.6653598754104156
Epoch 6/10, Val Loss: 0.06180254706897681, Val F1: 0.6272738866970925
100%|██████████| 2802/2802 [00:13<00:00, 214.76it/s]
Epoch 7/10, Train Loss: 0.05608997342783657, Train F1: 0.6762852794217135
Epoch 7/10, Val Loss: 0.061606788699556346, Val F1: 0.6119955603751678
100%|██████████| 2802/2802 [00:12<00:00, 219.87it/s]
Epoch 8/10, Train Loss: 0.054773784102306974, Train F1: 0.7028168750959859
Epoch 8/10, Val Loss: 0.060918421071346915, Val F1: 0.6596929681584783
100%|██████████| 2802/2802 [00:12<00:00, 219.59it/s]
Epoch 9/10, Train Loss: 0.053452545401376325, Train F1: 0.7213841483237264
Epoch 9/10, Val Loss: 0.06276448994088445, Val F1: 0.673005299366294
100%|██████████| 2802/2802 [00:12<00:00, 222.20it/s]
Epoch 10/10, Train Loss: 0.052219075438627, Train F1: 0.7355769043093471
Epoch 10/10, Val Loss: 0.06250936375896363, Val F1: 0.690785486788952
```

دقت مدل GRU

```
Training GRU...
100%|██████████| 2802/2802 [00:13<00:00, 206.74it/s]
Epoch 1/10, Train Loss: 0.0802555855484593, Train F1: 0.5087159277633225
Epoch 1/10, Val Loss: 0.05913869043368416, Val F1: 0.6674925360968371
100%|██████████| 2802/2802 [00:13<00:00, 202.08it/s]
Epoch 2/10, Train Loss: 0.05826961445348931, Train F1: 0.7286282546252687
Epoch 2/10, Val Loss: 0.05516813635296046, Val F1: 0.7338188228606713
100%|██████████| 2802/2802 [00:13<00:00, 205.86it/s]
Epoch 3/10, Train Loss: 0.053187685938542714, Train F1: 0.7660574392660094
Epoch 3/10, Val Loss: 0.05475560350665097, Val F1: 0.7604820623991068
100%|██████████| 2802/2802 [00:13<00:00, 207.02it/s]
Epoch 4/10, Train Loss: 0.049481756500913084, Train F1: 0.7915278655043556
Epoch 4/10, Val Loss: 0.055463121964804796, Val F1: 0.7560710725406433
100%|██████████| 2802/2802 [00:13<00:00, 203.22it/s]
Epoch 5/10, Train Loss: 0.04621113394470161, Train F1: 0.815012758250875
Epoch 5/10, Val Loss: 0.057059924565946496, Val F1: 0.7656721853406022
100%|██████████| 2802/2802 [00:13<00:00, 208.47it/s]
Epoch 6/10, Train Loss: 0.0430829816841541, Train F1: 0.8328699004253816
Epoch 6/10, Val Loss: 0.058244712953688244, Val F1: 0.749586237623993
100%|██████████| 2802/2802 [00:13<00:00, 205.41it/s]
Epoch 7/10, Train Loss: 0.04010744272052529, Train F1: 0.8495238756870073
Epoch 7/10, Val Loss: 0.06009359280718291, Val F1: 0.7533841402370911
100%|██████████| 2802/2802 [00:13<00:00, 204.94it/s]
Epoch 8/10, Train Loss: 0.037150966392751575, Train F1: 0.8689026752877889
Epoch 8/10, Val Loss: 0.06280235960526916, Val F1: 0.7600853466909664
100%|██████████| 2802/2802 [00:13<00:00, 206.56it/s]
Epoch 9/10, Train Loss: 0.03457561682651802, Train F1: 0.8822301383403511
Epoch 9/10, Val Loss: 0.06522025748524023, Val F1: 0.7586554159977084
100%|██████████| 2802/2802 [00:13<00:00, 207.98it/s]
Epoch 10/10, Train Loss: 0.03235156361800057, Train F1: 0.8917873137037917
Epoch 10/10, Val Loss: 0.06744814779046131, Val F1: 0.7500603031125992
```

دقت مدل LSTM

```
Training LSTM...
100%|██████████| 2802/2802 [00:13<00:00, 200.19it/s]
Epoch 1/10, Train Loss: 0.08777200605329405, Train F1: 0.4120829244991367
Epoch 1/10, Val Loss: 0.06280120749414478, Val F1: 0.5041659468408394
100%|██████████| 2802/2802 [00:14<00:00, 198.70it/s]
Epoch 2/10, Train Loss: 0.06049640016094499, Train F1: 0.6357658635633039
Epoch 2/10, Val Loss: 0.05718766626889968, Val F1: 0.6847698991603394
100%|██████████| 2802/2802 [00:13<00:00, 200.60it/s]
Epoch 3/10, Train Loss: 0.05448540471837212, Train F1: 0.731822219111005
Epoch 3/10, Val Loss: 0.05654359110943261, Val F1: 0.7284021518478238
100%|██████████| 2802/2802 [00:14<00:00, 196.27it/s]
Epoch 4/10, Train Loss: 0.05072663937376907, Train F1: 0.7747754373806292
Epoch 4/10, Val Loss: 0.055801178842846744, Val F1: 0.7376260308944961
100%|██████████| 2802/2802 [00:14<00:00, 199.64it/s]
Epoch 5/10, Train Loss: 0.04740983147545053, Train F1: 0.7959453679534958
Epoch 5/10, Val Loss: 0.05595968348823073, Val F1: 0.759336111954721
100%|██████████| 2802/2802 [00:13<00:00, 201.59it/s]
Epoch 6/10, Train Loss: 0.04438560457666173, Train F1: 0.8157288512402187
Epoch 6/10, Val Loss: 0.05779246886934389, Val F1: 0.7582640071702335
100%|██████████| 2802/2802 [00:13<00:00, 200.44it/s]
Epoch 7/10, Train Loss: 0.04186112995323945, Train F1: 0.8248381695321614
Epoch 7/10, Val Loss: 0.058813617082469474, Val F1: 0.7415537601889873
100%|██████████| 2802/2802 [00:13<00:00, 201.13it/s]
Epoch 8/10, Train Loss: 0.03949050213623098, Train F1: 0.8442827325372935
Epoch 8/10, Val Loss: 0.060437245073910274, Val F1: 0.7534200790620821
100%|██████████| 2802/2802 [00:14<00:00, 197.65it/s]
Epoch 9/10, Train Loss: 0.037327788458428456, Train F1: 0.8560997782223991
Epoch 9/10, Val Loss: 0.06138204489588823, Val F1: 0.7580518799450505
100%|██████████| 2802/2802 [00:14<00:00, 196.82it/s]
Epoch 10/10, Train Loss: 0.03541036680683781, Train F1: 0.8646629758022378
Epoch 10/10, Val Loss: 0.06336758097104951, Val F1: 0.7557772547266212
```

محمد حقیقت - 403722042