Final Project : Sentiment Analysis On The Snappfood Dataset

Zahra Tebyanian , Sadaf Fatollahy , Saeed Ghiasi

۲۷ دی ۱۴۰۲

نام استاد: دکتر فراهانی
نام درس : یادگیری ماشین پیشرفته

فهرست مطالب

۱.۰ مقدمه

Sentiment Analysis یا تجزیه و تحلیل احساسات یکی از فنون پرکاربرد در حوزه پردازش زبان طبیعی است که به بررسی و تحلیل نظرات، دیدگاه‌ها و احساسات موجود در متن‌ها می‌پردازد. در مورد داده‌های Snappfood، که یک سرویس سفارش آنلاین غذا است، تحلیل احساسات می‌تواند ارزش شرکت و کسب و کار را زیاد کند .

هدف اصلی Sentiment Analysis در داده‌های Snappfood، شناخت و درک نظرات و تجربیات کاربران درباره سرویس ارائه شده است. با تحلیل احساسات کاربران، شرکت می‌تواند از نقاط قوت و ضعف سرویس خود آگاه شده و اقداماتی را برای بهبود کیفیت خدمات و رضایت کاربران انجام دهد.

برای انجام Sentiment Analysis در داده‌های Snappfood، ابتدا نیاز است که داده‌ها را جمع‌آوری کرده و آن‌ها را بررسی کنیم. داده‌ها ممکن است شامل نظرات کاربران درباره غذاها، سرویس دهی، زمان تحویل، کیفیت غذا و عوامل دیگر باشد.

در این پروژه به طور کامل مراحل انجام Sentiment Analysis از جمله پیش پردازش، تبدیل بردار به ویژگی و روش های به کار برده شده را بر روی داده‌های Snappfood بیان میکنیم. در این پروژه از الگوریتم های Logistic regression، Random forest و LSTM استفاده کردیم.

۲.۰ کتاب خانه های مورد نیاز

پکیج های اصلی مورد استفاده ما در این پروژه موارد زیر است که ما اصلی ترین انها را که تا به حال از انها استفاده نکرده ایم را توضیح میدهیم :

- hazm: این پکیج یک کتابخانه پردازش زبان طبیعی برای زبان فارسی است. این پکیج شامل ابزارها و مدل‌هایی برای پیش‌پردازش متون فارسی مانند توکن‌بندی، حذف علائم نگارشی، استخراج پیشوندها و پسوندها، stem برداری و سایر وظایف مربوط به NLP است.
- re: پکیج re در زمینه پردازش زبان طبیعی ابزارها و توابعی را فراهم می‌کند که به وسیله آن‌ها می‌توانید با استفاده از Regular Expressions الگوهای مشخصی را در متن‌ها تشخیص داده و عملیاتی مانند جستجو، جایگزینی و استخراج اطلاعات را انجام دهید.
- urlextract: این پکیج یک کتابخانه است که برای استخراج URL ها (نشانی‌های وب) از متن‌ها استفاده می‌شود.
- emoji: این پکیج یک کتابخانه برای کار با شکلک‌ها و ایموجی‌ها در پایتون است که برای تشخیص، حذف یا تبدیل شکلک‌ها در رشته‌های متنی استفاده می‌شود.
- TfidfVectorizer: این پکیج برای داده های متنی بردار representation عددی ایجاد میکند.
...

۳.۰ درباره دیتاست

این دیتاست شامل سه بخش train ،validation و test میباشد که :

- تعداد داده های train برابر با 56700
- تعداد داده های validation برابر با 6300

- تعداد داده های test برابر با 7000

میباشند و دارای سه ستون با نام های label،comment و label id است که:

- comment: بیانگر نظرات مشتریان
- label: بیانگر لیبل مربوط به کامنت میباشد که در این دیتاست به دو نوع HAPPY و SAD تقسیم بندی شده است.
- label id : بیانگر ایدی مرتبط با لیبل است به طوری که لیبل HAPPY برابر 0 و لیبل SAD برابر 1 است.
- unnamed:0:تمام مقادیر ان Noneبود به همین دلیل از ابتدا ان را حذف کردیم.

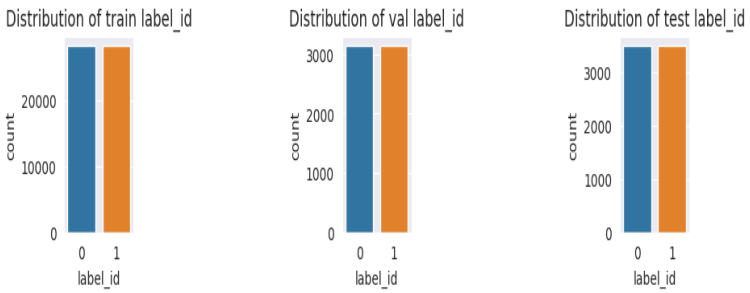
۴.۰ EDA and Pre processing

برای پیش پردازش داده های متنی کارهای زیر را انجام دادیم:

۱. با بررسي مشخص شد که دیتاست هیچ missing value اي ندارد.

۲. با بررسي مشخص شد که دیتاست هیچ dublicate value اي ندارد.

۳. در نمودار زیر مشخص شد دیتاست بالانس است.



۴. تابعي نوشتیم که ایموجی های موجود در متن را استخراج کرده و سپس انها ها را توصیف میکند .

۵. تابعي نوشتیم که اعداد فارسی را به انگلیسی تبدیل میکند.

۶. تابعي نوشتیم که حروفی را که ممکن است به کیبورد عربی نوشته شده باشد را به فارسی تبدیل کند.

۷. تابع اخر موارد زیر را انجام میدهد:

- همه URL های موجود در متن را استخراج کرده و به توکن url تغییر دادیم.
- همه ی smiley های موجود در متن را بدون توجه به بار معنایی انها به توکن smiley تغییر دادیم.

- حروف بزرگ انگلیسی در URL ها را به حروف کوچک تغییر دادیم.
- فاصله های اضافی را حذف کردیم.
- به جای کاراکترهای خاص و punctuation مثل علامت تعجب ، علامت سوال و ... space گذاشتیم.
- elongation موجود در متن را اصلاح کردیم.
- کامنت های فینگلیش را به None تبدیل کردیم سپس انها را حذف کردیم.

در نهایت تمام این تغییرات را در یک دیتاست جداگانه برای راحتی دخیره کردیم.

Model building ۵.۰

براي ساخت مدل ابتدا مجموعه داده‌های ورودی و برچسب‌ها را برای train،validationو test جداسازی کردیم. از آنجا که ورودي به يك مدل يادگيري ماشين بايد عدد باشد ما از TF-IDF براي عددي سازي داده هاي متني استفاده مي‌كنيم.

TF-IDF ۱.۵.۰

TF-IDF یک روش محاسبه وزنی است که در NLP استفاده می‌شود. این روش بر اساس تکرار و فراوانی کلمات در یک متن کار می‌کند و به ما اطلاعاتی درباره اهمیت یک کلمه در متن را می‌دهد.به طور کلی،TF-IDF از دو عامل ترکیبی استفاده می‌کند:

تکرار فراوانی کلمه در متن Term Frequency - TF: این مقدار نشان می‌دهد چقدر یک کلمه در یک متن خاص تکرار شده است. معمولا از روش‌های مختلفی برای محاسبه مقدار TF استفاده می‌شود، اما روش معمول ترین آن استفاده از تعداد تکرار کلمه در متن به تعداد کل کلمات متن است.

فراوانی معکوس متن در مجموعه متن ها Inverse Document Frequency - IDF: این مقدار نشان می‌دهد که چقدر کلمه در کل مجموعه متن ها رایج است یا در چند متن از مجموعه وجود دارد. محاسبه مقدار IDF به صورت لگاریتمی از تعداد کل متن ها تقسیم بر تعداد متنها که کلمه را شامل می‌شوند، انجام می‌شود. با استفاده از TF-IDF، متن‌ها به بردارهای عددی تبدیل می‌شوند که قابل استفاده در الگوریتم‌های یادگیری ماشینی هستند. این بردارها representation اي از محتواي متني را در يک فضاي چند بعدي ممکن می‌سازند، که می‌توان با استفاده از آنها شباهت میان متن‌ها را محاسبه کرد و به آنها مرتبه‌بندی داد. این روش را طوري تنظيم كرديم كه هر كلمه حداقل در ۲ جمله ظاهر شده باشد و در نهايت تعداد كل فيچر ها بايد حد اكثر 10000 تا شود.

Logistic regression and Random forest ۲.۵.۰

با توجه به اینکه مساله ما از نوع طبقه بندی دو کلاسه است در قسمت الگوریتم های یادگیری ماشین سنتی از این دو روش استفاده کردیم:

- رگرسیون لوجستیک یک الگوریتم یادگیری ماشینی است که برای مسائل دسته‌بندی بکار می‌رود. این الگوریتم با استفاده از تابع لوجستیک، احتمال لیبل کلاس را بر اساس ویژگی‌ها محاسبه می‌کند و سپس بر اساس این احتمال، نمونه‌ها را به یکی از دسته‌های مشخص تخصیص می‌دهد.

- الگوریتم Random Forest یک الگوریتم یادگیری ماشینی است که برای مسائل دسته‌بندی و رگرسیون استفاده می‌شود. این الگوریتم از مجموعه‌ای از درخت‌های تصمیم تشکیل شده است. در الگوریتم رندوم فارست، ابتدا n درخت تصمیم تصادفی ساخته می‌شود. این انتخاب تصادفی شامل انتخاب تصادفی ویژگی‌ها و نمونه‌ها از مجموعه داده آموزشی است. سپس هر درخت تصمیم به طور جداگانه آموزش داده می‌شود.

LSTM ۳.۵.۰

از آنجا که در شبکه عصبی های عادی ترتیب جملات در نظر گرفته نمی‌شود، از شبکه های عصبی LSTM استفاده کردیم. در LSTM برخلاف شبکه عصبی feed forward، ورودی می‌تواند به صورت دنباله باشد. این ویژگی باعث شده LSTM ها برای پردازش داده‌های زمانی بسیار مناسب باشند زیرا الگوها در داده‌های سری زمانی می‌توانند در فواصل مختلف واقع شوند.

Tokenization

برای دادن ورودی به شبکه عصبی LSTM ابتدا لازم است که داده‌های متنی‌مان را token بندی کنیم. برای اینکار از کلاس Tokenization کتابخانه keras استفاده کرده ایم. شئ ای از این کلاس با ورودی num_wordrs برابر ۲۰۰۰۰ ساخته شده است به این معنی که ۲۰ هزار از کلمات رایج در مجموعه داده train باقی می‌مانند و بقیه‌ی کلمات کمتر تکرار شده در نظر گرفته نمی‌شوند. سپس توکن‌ها را به دنباله‌هایی از اعداد تبدیل کرده و در آخر با یک متد padding از هم‌سایز بودن دنباله‌های ساخته شده در مجموعه‌های ورودی train، validation و test اطمینان حاصل می‌نماییم.

Architecture

معماری ما شامل ۲ hidden layer با تابع فعالیت relu و softmax در لایه آخر است. برای تابع هزینه از categorical cross entropy استفاده کردیم زیرا مساله از نوع classification است و این تابع هزینه برای زمانی مناسب است که هر نمونه ورودی دقیقاً به یک کلاس از چندین کلاس تعلق دارد. تعداد ایپاک ها را برابر ۱۰ گرفتیم ولی بعد از مشاهده overfitting از تکنیک EarlyStopping استفاده کردیم و مدل بعد از ۵ ایپاک متوقف شد.

Evaluation ۶.۰

برای ارزیابی عملکرد مدل ها همانطور که خواسته شده از روش Weighted F1-score استفاده کردیم. این روش متریکی است که presicion و recall را ترکیب میکند تا میزان عملکرد مدل را بسنجد و بیشتر برای زمانی مناسب است که دیتا بالانس نباشد و همچنین مساله از نوع طبقه بندی چندکلاسه باشد. در نهایت در جدول زیر نتیجه ارزیابی را ارائه کردیم که همه مدل ها تقریبا نتیجه برابری داشتند اما Logistic regression با اندکی تفاوت بهترین عملکرد را داشت. برای بهبود عملکرد سعی کردیم از نرمالیزیشن نیز استفاده کنیم اما به دلیل حجم بالای دیتاست لپتاپ قادر به محاسبه نبود و برای هر بار ران کردن حداقل ۱۲ ساعت زمان نیاز داشت. به همین دلیل از انجام آن خودداری کردیم.

	Weighted F1-Score
Logistic regression	0.857
Random Forest	0.853
LSTM	0.846

جدول ۱: Evaluation table

برای اطمینان از نتیجه کار به هر مدل یک کامنت به عنوان ورودی دادیم:

- کامنت ورودی به Logistic regression : دیگه از این رستوران غذا سفارش نمیدم. لیبل خروجی برابر یک بود پس مدل درست پیش بینی کرده است.
- کامنت ورودی به Random forest : دیگه از این رستوران غذا سفارش نمیدم. لیبل خروجی برابر یک بود پس مدل درست پیش بینی کرده است.
- کامنت ورودی به LSTM :غذا اصلا مزه خوبی نداشت. لیبل خروجی 88 درصد به یک نزدیک بود پس مدل درست پیش بینی کرده است.