دانشگاه صنعتی امیرکبیر

دانشکده‌ی علوم‌کامپیوتر

گردآورنده:

شیده هاشمیان

شماره دانشجویی:

۹۶۱۳۴۲۹

تمرین اول مباحثی در علوم کامپیوتر

عنوان : تحلیل احساسات جریان داده‌های میکروبلاگ

پاییز ۹۹

این پیاده‌سازی متشکل از چهار فایل **py.** است که هریک از آن‌ها و توابع موجود در آن‌ها در زیر توضیح داده شده است.
و برای اجرای برنامه، آدرس محل airline-train.csv، airline-dev.csv و ailine-test.csv را در given_data_root_path که در فایل constant.py وجود دارد وارد کنید.

۱. **ثابت‌ها** (constant.py):

این فایل شامل متغییرهایی است که در دیگر فایل‌ها مورد استفاده قرار می‌گیرند و در میان آن‌ها یکسان است که شامل آدرس پیکره‌های اولیه و regexهای استفاد شده برای نرمال‌سازی داده‌های متنی توییت‌ها (گرفته شده از توکنایزر معرفی شده در کلاس این درس) برای پردازش احساساست قرار دارند. همچین مجموعه‌ای از علائم غیر الفبایی که در پیکره‌ی آموزش موجود بود هست که در مرحله‌ی نرمال‌سازی متن از آن‌ها استفاده شود.

۲. **ابزارهای پردازش زبان(LP_toolkits.py):**

این فایل شامل یک متغییر سراسری (حاوی ایستواژه ها) است.

- **تابع tokenize(text)**: رشته‌ی توییت را ورودی می‌گیرد و پس از استفاده از regexهای موجود در فایل constants.py و استواژه‌های گرفته شده از کتابخانه‌ی nltk متن توییت‌ها را نرمال‌سازی می‌کند و آرایه‌ی از توکن‌های توییت بازمی‌گرداند.

۳. **دسته‌بند (classifier.py):**

این فایل شامل ۱۰ تابع است که در زیر توضیح داده شده‌اند.

- **تابع load_dataset_basic(csv_file_address)**: آدرس (همراه با اسم) فایل csv را می‌گیرد و تنها قسمت‌های مورد نیاز آن را برای آموزش مدل با خصوصیات پایه (’text‘ و ’airline_sentiment‘) را به‌صورت دیکشنری باز می‌گرداند.
- **تابع load_dataset_advance(csv_file_address)**: آدرس (همراه با اسم) فایل csv را می‌گیرد و تنها قسمت‌های مورد نیاز آن را برای آموزش مدل با خصوصیات پایه (’text‘ ، ’airline_sentiment‘ ، 'airline_sentiment:confidence'
و ’retweet_count‘) را به‌صورت دیکشنری باز می‌گرداند.
- **تابع extend_classes(advance_tweets_data_dict)**: خروجی تابع load_dataset_advance را دریافت می‌کند و با استفاده از خصوصیاتی که مستقل از متن توییت هستند دسته‌ها را از "مثبت"، "خنثی" و "منفی" به پنج دسته‌ی "بسیارمثبت"، "مثبت"، "کم‌مثبت"، "خنثی"، "کم‌منفی"، "منفی" و " بسیارمنفی " گسترش می‌دهد به این صورت که برای کلاس‌های منفی و مثبت ابتدا مقادیر ’retweet_count‘ و’airline_sentiment:confidence‘ را به اعداد بین ۰ تا ۳ map می‌کند و سپس میانگین وزن دار (با وزن بیشتر’airline_sentiment:confidence‘) می‌گیرد و باتوجه به مقدار بدست آمده به سه دسته تقسیم می‌کند. برای کلاس‌های خنثی تغییری ایجاد نمی‌کند. پس از اعمال این تغییرات را به‌صورت دیکشنری‌ای با دو کلید ’text‘ و ’airline_sentiment‘ (مشابه خروجی تابع load_dataset_basic) بازمی‌گرداند.

- **تابع chi_square_calculator(tweets_data_dic)**: با گرفتن دیکشنری‌ای به حالت خروجی‌های توابع extend_classes و load_dataset_basic ابتدا کلاس‌ها و ترم‌ها را index می‌کند و سپس با استفاده از آن‌ها و داده‌های متنی و کلاس‌های آن، جدول contingency را برای تمامی داده‌ها و کلاس‌ها می‌سازد. حال با استفاده از آرایه‌ی ساخته‌شده مقادیر χ^2 را برای هر ترم و کلاس محاسبه و در خانه‌ی متناظر با آن می‌نویسد. در نهایت آرایه‌ای از χ^2ها را بازمی‌گرداند. (آرایه‌ها از جنس numpy array هستند)
- **تابع vectorize(features_dict, text_arr)**: یک دیکشنری که کلیدهای آن تاثییرگزارترین (features) کلمات استخراج شده با استفاده از χ^2 است و ارزش‌های آن (values) indexهای تعلق گرفته به آن‌ها است همراه با text_arr که آرایه‌ای از متون توییت‌ها است ورودی گرفته. سپس از دیکشنری گفته شده به عنوان لغت‌نامه برای بردار سازی استفاده می‌شود، به این صورت که هنگام ساخت یک شی از کلاس CountVectorizer آن را به‌عنوان لغت‌نامه ورودی می‌دهد و سپس با استفاده از آن بردارهای شمارش را برای مجموعه‌ی داده متنی ساخته و آن را خروجی می‌دهد
- **تابع train_model(features_dict, tweets_data_dict, advance)**: در ای تابع features_dict مانند دیکشنری توضیح‌داده‌شده در تابع قبل است، tweets_data_dict هم مانند خروجی‌های توابع extend_classes و load_dataset_basic است و advance هم دو حالت true و False را می‌تواند اخذ کند که حالت استخراج خصوصیت‌ها را نشان می‌دهد. این تابع با ساخت یک شی از کلاس از SVC کتاب‌خانه‌ی sklearn به عنوان یک مدل SVM خطی، با استفاده از دیکشنری داده‌ها، ابتدا از تابع vectorize استفاده کرده و داده را بردار کرده، سپس با استفاده از تابع fit این کلاس و ورودی بردار متن‌ها و دسته‌های آن (به‌عنوان ورودی) مدل را آموزش می‌دهد و نهایتا آن به‌عنوان خروجی بازمی‌گرداند.
- **تابع basic_classifier_training(train_csv_file_address, develop_csv_file_address)**: این تابع توابع قبلی را به‌ترتیب برای آموزش مدل با استفاده از فایل airline-train.csv بکار می‌گیرد. ابتدا با استفاده از load_dataset_basic داده‌های مورد نیاز در این حالت را جدا کرده و این خروجی را به تابع chi_square_calculator داده تا χ^2 برای آن محاسبه شود. سپس از ماتریس χ^2 استفاده و داده‌های airline-dev.csv استفاده کرده تا بهترین features ها را استخراج و در نهایت بهترین مدل و بهترین freatures را در فایل‌های features_basic.pickle و svm_basic_model.pickle ذخیره می‌کند. (به این صورت که مقادیر موجود در ماتریس χ^2 را مرتب کرده و هر بار بین $\frac{1}{20}$ تا $\frac{20}{20}$ آن‌ها را برای این که کلمه‌های متناظر با آن‌ها به عنوان feature درنظر گرفته شوند انتخاب کرده، مدل را آموزش داده و با استفاده از داده‌ی توسعه آن را سنجیده و بهترین حالت را ذخیره می‌کند.)

-

- **تابع (advance_classifier_training(train_csv_file_address, develop_csv_file_address**: این تابع توابع قبلی را به‌ترتیب برای آموزش مدل با استفاده از فایل airline-train.csv بکار می‌گیرد. ابتدا با استفاده از load_dataset_advance داده‌های مورد نیاز در این حالت را جدا کرده و این خروجی را به تابع extend_classes داده تا تعدا کلاس‌ها گسترش پیدا کنند و سپس خروجی آن را به تابع chi_square_calculator داده تا χ^2 برای آن محاسبه شود. سپس از ماتریس χ^2 و داده‌های airline-dev.csv استفاده کرده تا بهترین features ها را استخراج و در نهایت بهترین مدل و بهترین freatures را در فایل‌های features_advance.pickle و svm_advance_model.pickle ذخیره می‌کند. (به این صورت که مقادیر موجود در ماتریس χ^2 را مرتب کرده و هر بار بین $\frac{1}{20}$ تا $\frac{20}{20}$ آن‌ها را برای این که کلمه‌های متناظر با آن‌ها به عنوان feature درنظر گرفته شوند انتخاب کرده، مدل را آموزش داده و با استفاده از داده‌ی توسعه آن را سنجیده و بهترین حالت را ذخیره می‌کند.)
- **تابع (evaluation(test_data_file_add, advance**: آدرس (همراه با اسم) فایل csv داده‌های آزمون همرا با حالت مدلی که برای ارزیابی مورد استفاده قرار می‌گیرد را ورودی گرفته، سپس ابتدا با استفاده از یکی از توابع load_dataset_basic و یا load_dataset_advance (متناسب با حالت انتخابی) متن داده‌ها را برای بردار شدن آماده می‌کند (برای حالت advance = True خروجی این تابع، به تابع extend_clasees داده‌شده تا کلاس‌های آن مناسب با کلاس‌های مدل پیشرفته شوند و از خروجی این تابع برای مراحل بعدی استفاده می‌شود)، سپس با استفاده فایل pickle ذخیره شده برای features و تابع vectorize داده‌های متنی را به‌صورت بردار در آورده. حال با استفاده از این بردار و مدل ذخیره شده، کلاس هر یک را تخمین می‌زند و سپس با مقایسه‌ی کلاس‌های داده شده و کلاس‌های تخمین‌زده شده، معیارهای خواسته شده برای ارزیابی مدل را محاسبه و چاپ می‌کند.
- **تابع (predict(tweet, advance)**: متن توییت همرا با حالت مدلی که برای تخمین مورد استفاده قرار می‌گیرد را ورودی گرفته، سپس ابتدا با استفاده از تابع tokenize متن توییت را برای بردار شدن آماده می‌کند سپس با استفاده فایل pickle ذخیره شده برای features و تابع vectorize داده‌ی متنی را به‌صورت بردار درآورده. حال با استفاده از این بردار و مدل ذخیره شده، کلاس این توییت را تخمین می‌زند و آن را به‌عنوان خروجی بازمی‌گرداند.

۴. فایل اجرایی (main.py):

این فایل در یک حلقه‌ی بی‌نهایت (تا زمان خروج توسط اجرا کننده) گزینه‌هایی به اجرا کننده داده و متناسب با انتخاب آن‌ها خروجی مرتبط می‌دهد. این گزینه‌ها مشاهده‌ی نتایج ارزابی و گرفتن دسته‌ی متناسب با توییت وارد شده برای هر دو حالت آموزش (ساده و پیشرفته) است.

۵. نتایج آزمون

Basic Feature Extraction

	Negative	Neutral	Positive
Confusion matrix	767 239 335 1587	2051 308 241 328	2335 156 127 310
Precision	0.87	0.52	0.67
Recall	0.83	0.58	0.71
F1	0.85	0.54	0.69
Error rate	0.20	0.19	0.10

- avg precision: 0.6834
- accuracy: 0.8400
- f1 macro: 0.6926
- f1 micro: 0.7400

Advance Feature Extraction

	High-negative	negative	Low-negative	neutral	Low-positive	Positive	High-positive
Confusion matrix	2776 124 24 4	728 190 516 1494	2914 14 0 0	2111 338 181 298	2914 12 2 0	2370 151 116 291	2914 12 2 0
Precision	0.03	0.89	0.00	0.47	0.00	0.66	0.00
Recall	0.14	0.74	0.00	0.62	0.00	0.72	0.00
F1	0.05	0.81	0.00	0.53	0.00	0.69	0.00
Error rate	0.05	0.24	0.00	0.18	0.00	0.09	0.00

- avg precision: 0.2922
- accuracy: 0.9180
- f1 macro: 0.2972
- f1 micro: 0.713

به‌طور کلی می‌توان دید که هر دو مدل برای توییت‌های خنثی عملکرد خوبی ندارند (باتوجه به معیار F1 که برای ارزیابی کلی مدل بهتر است) و همچنین درصد خطای هر دو مدل برای داده‌های منفی از دیگر داده‌ها بیشتر است.

همچنین با مقایسه‌ی دقت دو مدل می‌توان دید که دقت مدل در حالت دوم بیشتر است.

و باتوجه به confusion matrix در حالت دوم می‌توان دید که اثر داده‌های آموزش (و همچنین آزمون) به‌طور خیلی قطعی یا خیلی غیرقطعی در یکی از کلاس‌های منفی یا مثبت قرار ندارند و اکثرا یک نسبت قطعیت را دارند.