دانشگاه صنعتی امیرکبیر

دانشکده‌ی علوم‌کامپیوتر

گردآورنده:

شیده هاشمیان

شماره دانشجویی:

۹۶۱۳۴۲۹

تمرین دوم درس پردازش زبان طبیعی

عنوان : دسته‌بندی متن

پاییز ۹۹

این پیاده‌سازی متشکل از پنج فایل py. است که هریک از آن‌ها و توابع موجود در آن‌ها در زیر توضیح داده شده است. و برای اجرای برنامه، آدرس محل train.csv و valid.csv را در given_data_root_path که در فایل constant.py وجود دارد وارد کنید. همچنین تمام فایل‌های ساخته شده در برنامه، در پوشه‌ای که آدرس آن رشته‌ی document_root_path (در فایل constant.py) است قرار دارند.

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، آدرسی که برنامه فایل‌هایی که در طول اجرا تولید می‌کند در آن آدرس ذخیره کند هست. همچین مجموعه‌ای از علائم نگارشی و علائم غیر الفبایی (به‌غیر از نقطه، علامت سوال و اعداد) که در پیکره‌ی آموزش موجود بود هست که در مرحله‌ی نرمال‌سازی متن از آن‌ها استفاده شود. علاوه بر این‌ها شامل ثابت عددی محدودسازی مجموعه لغت که در دیگر بخش‌ها مورد استفاده قرار می‌گیرد هست.

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

این فایل متشکل از سه تابع است.

- تابع sub_alphabets که برگرفته شده از تابعی با همین نام در پکیج parsivar هست باتوجه به نیاز در این برنامه در برخی از قسمت‌ها عوض شده است که با گرفتن یک رشته در آن تمام حروفی که در این برنامه برای ما معنی‌دار هستند را به یک مجموعه حروف مشخص map می‌کند تا کلماتی که یک نگارش دارند یکسان شناسایی شوند.
- تابع normalizer نرمال‌سازی ابتدایی که شامل اجرای تابع sub_alphabets بر روی رشته‌های ورودی، حذف علائم نگارشی و علائم غیر الفبایی (به‌غیر از نقطه، علامت سوال و اعداد) و تغییر اعداد به N را انجام داده و سپس رشته‌ی نهایی را به عنوان خروجی بازمی‌گرداند.

۳. **آماده‌سازی و شناخت داده (preprocessing.py):**

- تابع char_word_number(vocabulary_json_add): آدرس یک لغت‌نامه که بر اساس تعداد رخداد کلمات مرتب شده‌اند به عنوان ورودی می‌گیرد و تعداد کل کلمات و کاراکترهای متمایز را چاپ می‌کند.
- تابع most_frequent_words(vocabulary_json_add): آدرس یک لغت‌نامه که بر اساس تعداد رخداد کلمات مرتب شده‌اند به عنوان ورودی می‌گیرد و ۱۰۰۰۰ کلمه‌ای که بیشترین تکرار را داشته‌اند در فایل most_frequent.txt ذخیره می‌کند.
- تابع most_frequent_words_percentage_among_all(): درصد ۱۰۰۰۰ لغتی که بیتشرین تکرار را داشته‌اند به کل لغت‌ها در خروجی چاپ می‌کند. (با استفاده از لغت‌نامه‌ی ذخیره شده)
- تابع reformat_least_frequent_words(train_doc_arr, most_frequent_file_add): آرایه‌ای از داده‌ها که نرمال‌سازی شده اند و آدرس فایل most_frequent.txt را به عنوان ورودی می‌گیرد و لغاتی که پرتکرار نیستند را با 'UKN' جای‌گزین می‌کند و لیست خبرهای ویرایش شده را خروجی می‌دهد.

- تابع char_word_txt_constructor(most_frequent_file_add): آدرس فایل most_frequent.txt را می‌گیرد و براساس آن دو فایل words.txt و chars.txt را می‌سازد.
- تابع read_csv_train_data(doc_add): آدرس فایل csv داده‌های آموزش را به عنوان ورودی می‌گیرد، پس از نرمال‌سازی داده‌ها و ساختن most_frequent.txt و ویرایش متن خبرها آنهارا به‌صورت آرایه در یک فایل pickle ذخیره می‌کند، همچنین دسته‌ها آن‌ها را هم در فایلی جدا ذخیره می‌کند.

۴. طراحی دسته‌بند (classifier.py):

- تابع word_char_index_construction(most_frequent_file_add) باتوجه به این که کلمات در داده‌های متنی مورد پردازش کلمات موجود در فایل most_frequent.txt هستند و 'UKN'، برای index کردن کلمات و حروف تنها داده‌ی این کلاس کافی‌است؛ پس با استفاده از آن چهار لغت‌نامه index2word,word2index,index2char,char2index خواسته شده را می‌سازیم. همچنین با توجه به یکسان‌سازی طول جملات، 'PAD' را هم به‌عنوان یک حرف و هم به عنوان یک کلمه در ساخت این لغت‌نامه‌ها لحاظ می‌کنیم.
- توابع word_indexing(word2index, train_news_arr) و char_indexing(char2index, train_news_arr) باتوجه به نوع indexing آرایه‌ای از اخبار به‌همراه لغت‌نامه‌ی متناسب دریافت کرده و آرایه‌ای از اخبار index شده را خروجی می‌دهد.
- توابع handle_docs_length_word_level(news_arr) و handle_docs_length_char_level(news_arr) (باتوجه به نوع indexing) اخباری که طول کمتر از میانگین داره را حذف و مابقی را با اضافه کردن 'PAD' هم‌طول می‌کند و آرایه‌ای از اخبار ویرایش شده را خروجی می‌دهد.
- تابع clean(raw_news_arr): آرایه‌ای از اخبار را ورودی می‌گیرد و پردازش‌های اولیه شامل نرمال‌سازی و جای‌گزینی کلمات کم‌تکرار را انجام می‌دهد و آرایه‌ای از اخبار ویرایش شده را به‌عنوان خروجی می‌دهد.
- تابع tokenize(news_cleaned_arr, word2index, char2index, level): آرایه‌ای از اخبار، دو لغت‌نامه‌ی index ها و سطح را ورودی می‌گیرد. سپس باتوجه به سطح آموزش اخبار را به تابع handle_docs_length_word_level(news_arr) یا handle_docs_length_char_level(news_arr) داده و خروجی آن را با استفاده از و char_indexing یا word_indexing آرایه‌ای از اخبار index شده را می‌گیرد و خروجی می‌دهد.
- تابع count_vector_constructor(doc_indexed_arr, features_index): باتوجه به سطح آموزش لیست کلیدهای index2word و یا word2index را به‌عنوان feature_index همراه با آرایه‌ای از داده‌های index شده را ورودی می‌گیرد و count_vector اخبار را ساخته و خروجی می‌دهد. (از آن برای ساخت tf-ifd برای کلمات و حروف استفاده می‌شود).
- تابع sub_10_chunk(doc_indexed_arr): آرایه‌ای از اخبار index شده را ورودی می‌گیرد و به ۱۰ قسمت مساوی تقسیم می‌کند و به عنوان آرایه‌ای تو در تو، این اخبار را خروجی می‌دهد.
- تابع vectorize(word_indexed_doc_arr, char_indexed_doc_arr, index2word, index2char, level): آرایه‌ای از اخبار index شده، دو لغت‌نامه‌ی index2word و index2char همراه با عد سطح را ورودی می‌گیرد، ابتدا با استفاده از تابع قبل داده‌های متنی را به ۱۰ قسمت تقسیم کرده، سپس برای هر قسمت count_vector آن را محاسبه و به‌عنوان ورودی به تابعی از کلاس TfidfTransformer از کتاب‌خانه‌ی sklearn داده و بردار خروجی را به عنوان فایل train_x_word_level_{#}.pickle برای استفاده زمان آموزش ذخیره می‌کند و همین روند را برای

باقی دسته‌ها انجام می‌دهد. این کار به این منظور است که در صورت مشکل در استفاده از تمام داده‌ها برای آموزش در memory بتوان دسته دسته آموزش داد.

- تابع defining_model(): در این تابع با استفاده از کلاس SGDClassifier یک دسته‌بند SVM تعریف می‌کند و آن را به عنوان خروجی می‌دهد.
- تابع train(level) : سطح را ورودی می‌گیرد و متناسب با آن ۱۰ دسته بردارهای ذخیره شده در زمان اجرای تابع vectorize را می‌خواند و همچنین باتوجه به دسته‌های ذخیره شده زمان اجرای تابع csv را خوانده و متناسب با دسته‌ی تمامی اخبار به ۱۰ دسته تقسیم می‌کند. حال طی ۵ تکرار، هر بار ۲ دسته را کنار گذاشته و با استفاده ۸ دسته‌ی دیگر مدل را آموزش می‌دهد و با دو دسته‌ی کنار گذاشته شده امتیاز مدل را می‌سنجد و از بین ۵ مدل، مدلی که بهترین عملکرد را داشته است در فایلی با نام SVM_model_word_base.pickle و یا SVM_model_char_base.pickle (متناسب با سطح داده‌شده) ذخیره می‌کند.

۵. اعتبار سنجی (evaluation):

- تابع test_doc_prep(test_doc_add, index2word, word2index, index2char, char2index, level) : این تابع آدرس فایل آزمون(test.csv) و چهار دیکشنری و سطح را ورودی می‌گیرد و همان تغییراتی که داده‌ی آموزش قبل از داده شدن به دسته‌بند روی آن اعمال شد، روی این داده نیز اعمال می‌شود و نهاییتا دو فایل که یکی اخبار بردار شده و دیگری کلاس متناظر با آن‌ها است را به صورت فایل pickle ذخیره‌سازی می‌کند.
- تابع evaluation(test_doc_add, index2word, word2index, index2char, char2index, level): این تابع آدرس فایل آزمون(test.csv) و چهار دیکشنری و سطح را ورودی می‌گیرد و تابع قبلی را (در صورت وجود نداشتن فایل‌های ویرایش شده‌ی آزمون برای خواندن) اجرا می‌کند. پس از آن با خواندن برادار اخبار آزمون و استفاده از مدل ذخیره شده (متناسب با سطح)، دسته‌های این اخبار را تخمین می‌زند(y_pred). سپس فایلی که دسته‌های واقعی این اخبار را نشان می‌دهد(y_true) می‌خواند. حال با استفاده از این دو و استفاده از کتابخانه‌ی sklearn ، معیارهای خواسته شده را محاسبه و چاپ می‌کند. همچنین نمودار ROC را برای همه‌ی دسته‌بندی ها بر روی یک plot رسم می‌کند.

۶. تحلیل نتایج:

نتایج بدست آمده از اجرای Evaluation بر روی داده‌های آزمایش در فایل evaluation metrics.pdf قرار داده‌شده است.

طبق انتظار ۳ دسته‌ی "category"، "nan" و "فیلم و صوت" که فاقد اطلاعات متنی بودن، هیچ یک از اخبار آزمایش توسط مدل سطح کلمه در این دسته قرار نگرفته‌اند که با بررسی confusion matrix می‌توان این را دید اما تعدادی داده در حالت سطح حرف در یکی از آن‌ها قرار گرفته‌اندزیرا در این حالت فاصله را نیز به‌عنوان یک حرف در نظر گرفته شده و امکان ظهور آن در دسته‌های گفته شده وجود دارد.

همچنین طبق انتظار عملکرد سطح کلمه از عملکرد سطح حرف بهتر است زیرا داده‌های خیلی بیشتری در حالت کلمه در نظر گرفتن اخبار در اختیار داریم.